

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

**SIMULATION OF A MULTITARGET, MULTISENSOR,
TRACK-SPLITTING TRACKER FOR MARITIME
SURVEILLANCE**

by

Mark A. Olson

September 1999

Thesis Co-Advisor:
Thesis Co-Advisor:

Harold Titus
Herschel Loomis

Approved for public release; distribution is unlimited.

19991126 114

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1999		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Simulation of a Multitarget, Multisensor, Track-Splitting Tracker For Maritime Surveillance			5. FUNDING NUMBERS	
6. AUTHOR(S) Olson, Mark A.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This study adapts some established target tracking techniques for use in the maritime surface surveillance role and tests them with computer generated data. Computer simulation of a track splitting tracker capable of operating in this undersampled and asynchronous environment is presented. The tracker uses standard and extended Kalman Filter algorithms to estimate target state from latitude and longitude or line of bearing position measurements. Prior to state estimation, all measurements are processed to retain only those that meet feature and geographic gate thresholds. All measurements passing these criteria will update the target state and be scored based on a goodness of fit with the model. The state estimate with the best score is selected as the correct one for display purposes, while all state estimates continue to be processed with subsequent measurements. Several runs of the simulation are discussed here to illustrate the performance of track splitting and the effect of several key tracker parameters.				
14. SUBJECT TERMS Multitarget Tracking, Track-Splitting Tracker, Kalman Filters			15. NUMBER OF PAGES 96	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

**SIMULATION OF A MULTITARGET, MULTISENSOR, TRACK-SPLITTING
TRACKER FOR MARITIME SURVEILLANCE**

Mark A. Olson
Lieutenant, United States Navy
B.S., United States Naval Academy, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

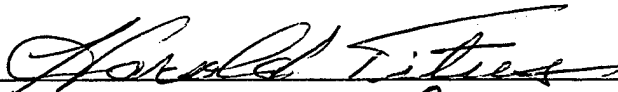
**NAVAL POSTGRADUATE SCHOOL
September 1999**

Author:

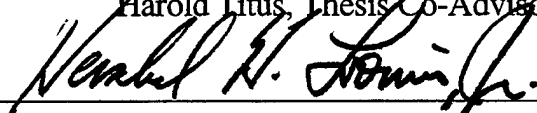


Mark A. Olson

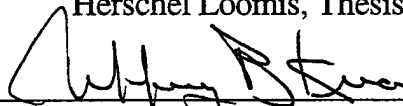
Approved by:



Harold Titus, Thesis Co-Advisor



Herschel Loomis, Thesis Co-Advisor



Jeffrey Knorr, Chairman

Department of Electrical and Computer Engineering

ABSTRACT

This study adapts some established target tracking techniques for use in the maritime surface surveillance role and tests them with computer generated data. Computer simulation of a track splitting tracker capable of operating in this undersampled and asynchronous environment is presented. The tracker uses standard and extended Kalman Filter algorithms to estimate target state from latitude and longitude or line of bearing position measurements. Prior to state estimation, all measurements are processed to retain only those that meet feature and geographic gate thresholds. All measurements passing these criteria will update the target state and be scored based on a goodness of fit with the model. The state estimate with the best score is selected as the correct one for display purposes, while all state estimates continue to be processed with subsequent measurements. Several runs of the simulation are discussed here to illustrate the performance of track splitting and the effect of several key tracker parameters.

TABLE OF CONTENTS

I. INTRODUCTION.....	1
A. BACKGROUND: THE MARITIME SURVEILLANCE ENVIRONMENT	1
B. THE TRACK SPLIT APPROACH	3
C. THESIS OUTLINE	4
II. CONTACT AND MEASUREMENT MODEL.....	7
A. CONTACT MOTION MODEL	7
B. MEASUREMENT MODEL.....	8
III. THEORETICAL BASIS FOR THE TRACK SPLITTING TRACKER.....	13
A. THE KALMAN FILTER.....	13
B. MEASUREMENT REJECTION USING FEATURES.....	16
C. ELLIPSOIDAL GATING.....	17
D. TRACK SPLITTING AND THE LIKELIHOOD FUNCTION.....	18
E. TRACK DROPPING CRITERIA.....	20
IV. COMPUTER SIMULATION.....	21
A. OVERVIEW.....	21
B. SIMULATION 1: FEATURE REJECTION TEST	23
C. SIMULATION 2: GATING TEST.....	34
D. SIMULATION 3: SCORING TEST	37
E. SIMULATION 4: COMBINED TEST	42
F. SIMULATIONS WITH EXTERNAL MEASUREMENT DATA	47
V. CONCLUSION.....	49
A. SUMMARY	49

B. FURTHER RESEARCH.....	50
APPENDIX	53
LIST OF REFERENCES	79
INITIAL DISTRIBUTION LIST	81

LIST OF FIGURES

Figure 1. Functional Flow of Track-Splitting Simulation	22
Figure 2. True Target Motion for Targets 1 and 2.	26
Figure 3. True Target Position and Measurements At Sample Times.....	27
Figure 4. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.	28
Figure 5. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times for Target 1.....	29
Figure 6. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times for Target 2.....	30
Figure 7. Simulation 2 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.	35
Figure 8. Simulation 3 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.	38
Figure 9. Simulation 3 True Target Position, Target Measurements, and Corrected Estimates at Sample Times. Best Estimate Only For Target 1.	39
Figure 10. Simulation 4 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.	44

LIST OF TABLES

Table 1. Statistics for Simulation 1: Feature Rejection Test.....	33
Table 2. Statistics for Simulation 2: Gating Test	36
Table 3. Statistics for Simulation 3: Scoring Test.....	40
Table 4. Statistics for Simulation 4: Combined Test	45

ACKNOWLEDGEMENTS

I am grateful for the advice and interest of my co-advisors in this project, Professor Hal Titus and Professor Herschel Loomis. By affording me a great deal of latitude in the design of this algorithm, the research was especially challenging and rewarding. Also, Dr. Alan Ross and Professor Gary Hutchins were helpful in their areas of expertise.

I appreciate the encouragement and interest of my wife, Debbie. Finally, the many hours of coding at my home computer were made more enjoyable by the companionship of my feline co-author, Spook, who unfortunately passed on before the final chapter was complete.

I. INTRODUCTION

A. BACKGROUND: THE MARITIME SURVEILLANCE ENVIRONMENT

The complexity of the target tracking problem varies according to a number of factors. Significant research has been devoted to the study of trackers in varying levels of target motion model, (plant) noise and measurement noise. Performance under a variety of target maneuvers has also been studied. Clutter environments have been added to the model, further complicating the tracking problem. Tracking problems containing multiple targets, multiple sensors, and target maneuvers are most complex, and extensive research has gone into developing algorithms for these situations. A maritime surveillance application for such a tracker is the focus of this study. The characteristics of this environment are unique in many respects, presenting an interesting and challenging tracking problem.

One of the most problematic aspects of maritime surveillance tracking is the long time interval between position updates. Typical air target tracking systems, such as a fire control system, will receive measurements of the target many times each second from tracking radars with high pulse repetition rates. A maritime surveillance tracking system, however, will receive measurements of contacts at intervals measured in minutes or hours depending on such factors as contact geographic location, asset availability, and source level or signal strength. Fortunately, because it is a surveillance system, the maritime surveillance tracker has a relaxed accuracy requirement when compared to an air target

tracker: accuracy within hundreds or a few thousand yards is acceptable in this application, but clearly unacceptable in air target tracking. Also, the relatively slow speed of a surface contact mitigates the effect of longer update intervals. Nonetheless, the undersampled environment is a significant challenge that a maritime surveillance tracker must overcome.

There are a variety of participants in the maritime surveillance effort: active radars from ships or aircraft at sea, nationally tasked sensors, shipping control officials using civilian shipping reports, to list some. While the sources or sensors from which a contact's position is obtained may be dissimilar (acoustic or electronic line of bearing, active radar, or a filed plan of intended movement), the data is integrated into the maritime surveillance picture via the Sensor Report. From this report, the tracker receives either a line of bearing or latitude and longitude for the target and a confidence region. This confidence region may be an ellipse (for latitude and longitude reports) or a bearing swath (for line of bearing reports) and is sensor dependent. Other information that may be contained in the sensor report includes estimated course and speed and feature data such as emitter frequency and pulse repetition frequency. Sensor reports allow dissimilar sensors to participate in the tracking effort.

Unlike many tracking problems where position information is received at regular intervals, there is no synchronization of sensor reports in maritime surveillance. The various sensors participating each operate according to their own reporting abilities and frequently have irregular reporting periods. Consequently, batch processing of measurements is impractical.

Finally, the number of tracks that a maritime surveillance tracker must handle to be of value further complicates the design. With the goal of providing situational awareness of the surface traffic in a geographic area to naval forces, a maritime surveillance tracker would have to support thousands of tracks. While advances in computing and memory storage technologies will enable the design of a high volume tracker, the number of contacts to be tracked in a geographic area will vary with time such that the tracking system is faced with an unknown number of targets. This restriction eliminates many tracking algorithms from consideration.

B. THE TRACK SPLIT APPROACH

The track split approach is a multi-target tracking algorithm that lends itself to several problems, including single target in clutter, multiple targets in clutter, and multiple targets in clutter with the number of targets unknown. Unlike some simple trackers which assume the measurement closest to the estimate is the correct one, track splitting trackers will use all measurements that fall within a validation region, or gate, to update the target state. A likelihood function is then used to eliminate unlikely sequences that presumably represent false tracks. In this way, the tracker postpones important decisions about which measurements originated from a target until additional measurement information is received.

The track split approach shares the Kalman Filter based prediction and update sequence used by other trackers. This widely used technique obtains the optimal estimate of the target state, for a given dynamic model, by recursively minimizing the mean square

error. The filter is tunable to a variety of applications and environments by properly selecting the dynamic and noise model. Asynchronous or missed measurements are readily handled by the filter and a measure of tracker accuracy (relative to the specified model) is always available in the form of the target's covariance matrix. Because of its versatility, a Kalman Filter based Track Splitting tracker can be developed for a maritime surveillance role.

The track split approach was selected for this study because it is the simplest algorithm capable of meeting the requirements imposed by the maritime surveillance environment. Track splitting is a technique for allowing the Kalman Filter to handle false measurements and multiple contacts. Furthermore, this approach does not require the number of targets to be known and will automatically initialize new tracks when more than one measurement falls within the gate of a currently held track. Although other algorithms are also capable of performing under these conditions (most notably, the Multiple Hypothesis Tracker (MHT)), the track split approach is a logical starting point in the design and evaluation of a maritime surveillance tracker.

C. THESIS OUTLINE

The remainder of this thesis is organized as follows: Chapter II will discuss the contact and measurement model by presenting the state space dynamic representation and noise model. Chapter III will detail the tracker algorithm, including a review of standard and extended Kalman Filter equations, track split logic, gating, and estimate error ellipses. Chapter IV will present and discuss the simulation parameters and results. Chapter V will

draw conclusions about this study and outline some areas for future research. The tracker simulation code, written for MATLAB®, is contained in the Appendix.

II. CONTACT AND MEASUREMENT MODEL

A. CONTACT MOTION MODEL

The tracker assumes contacts have a two-dimensional (x-y), second order (position and velocity) dynamic model. The model is implemented in discrete form according to:

$$\mathbf{x}_{k+1} = \mathbf{F}_k \cdot \mathbf{x}_k + \mathbf{v}_k \quad 2.1$$

where \mathbf{x} is the state space vector composed of x position, x velocity, y position, and y velocity. \mathbf{F} is the linear matrix:

$$\mathbf{F}_k = \begin{bmatrix} 1 & \Delta T_k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.2$$

Here, ΔT is the time interval between samples. Plant noise is included in the dynamic model to account for contact deviations from straight-line motion. \mathbf{v}_k is a zero mean Gaussian random variable with known covariance, \mathbf{Q} , given by:

$$\mathbf{Q} = q^2 \begin{bmatrix} \frac{\Delta T^3}{3} & \frac{\Delta T^2}{2} & 0 & 0 \\ \frac{\Delta T^2}{2} & \Delta T & 0 & 0 \\ 0 & 0 & \frac{\Delta T^3}{3} & \frac{\Delta T^2}{2} \\ 0 & 0 & \frac{\Delta T^2}{2} & \Delta T \end{bmatrix} \quad 2.3$$

The parameter q^2 is used to adjust the magnitude of plant noise. Properly selecting this parameter allows the designer to optimize tracker performance for the expected behavior of the contacts. Contacts that maintain relatively straight line motion, in open ocean transit for example, will not deviate greatly from the dynamic model. A low value for q^2 will give optimum tracker performance. Maneuvering contacts, however, conform to the model poorly. A large value for q^2 allows the tracker to maintain track.

B. MEASUREMENT MODEL

Sensors feeding into the maritime surveillance tracking system obtain contact positions as either single dimensional lines of bearing or two dimensional range and bearing measurements. Passive systems, such as acoustic hydrophones or electronic direction finding receivers, will produce lines of bearing. Active radars and electronic geolocation systems employing techniques such as Time Difference of Arrival measurements produce two-dimensional position reports. All sensors have an associated measurement error which is modeled as a zero mean Gaussian random variable.

Position reports contain the sensor's measurement, along with a metric of measurement uncertainty. In the case of Line of Bearing reports, a 95% confidence swath represents about 2.5 standard deviations of the measurement noise. Thus, σ_θ can be readily found from the sensor report. Latitude and Longitude reports will report measurement error statistics in the form of an error ellipse. This ellipse is related to the eigenvalues and eigenvectors of the sensor's covariance matrix, \mathbf{R} , (Kirk, 1975):

$$\frac{\mathbf{X}'^2}{\lambda_1} + \frac{\mathbf{Y}'^2}{\lambda_2} = c^2 \quad 2.4$$

where

λ_1 is the eigenvalue associated with the \mathbf{X}' eigenvector of \mathbf{R}

λ_2 is the eigenvalue associated with the \mathbf{Y}' eigenvector of \mathbf{R}

c^2 is a constant that determines the size of the ellipse

The relationship between c^2 and the probability of a measurement lying within the ellipse is tabulated for the two dimensional case (Kirk, 1975). In this study, c^2 is chosen to be 6, representing approximately 95% probability that the measurement falls within the error ellipse.

The tracker must invert the ellipse generating process to obtain the measurement covariance matrix. The sensor report contains the semi-major axis length (a), semi-minor axis length (b), and orientation of the ellipse (ϕ). The eigenvalues can be found by:

$$\lambda_1 = \frac{a}{2c}$$

$$\lambda_2 = \frac{b}{2c} \quad 2.5$$

The orientation of the semi-major and semi-minor axes may be expressed as unit vectors, giving the eigenvectors of the matrix that generated the ellipse directly. A similar

matrix to \mathbf{R} (one with the same characteristic equation) can be obtained as follows (Kaplan, 1981):

$$\mathbf{R}' = \mathbf{C} * \mathbf{B} * \mathbf{C}^{-1} \quad 2.6$$

where \mathbf{C} is composed of the eigenvectors obtained from the ellipse orientation:

$$\mathbf{C} = [\mathbf{X}' \quad \mathbf{Y}'] \quad 2.7$$

and \mathbf{B} is composed of the eigenvectors obtained from the semi-major and semi-minor axes:

$$\mathbf{B} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad 2.8$$

While the matrix \mathbf{R}' is not necessarily identical to the sensor's covariance matrix, it has the same characteristic equation and is therefore acceptable for use in the tracker.

A coordinate transformation from polar (range and bearing) to cartesian (latitude and longitude) occurs at the sensor. For purposes of this simulation, the cartesian position in distance units is transformed into latitude and longitude coordinates using a flat earth model, with one degree equal to 60 nautical miles (nm). A nonlinear rotation is used to obtain the measurement in cartesian:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} r_m \sin \theta_m \\ r_m \cos \theta_m \end{bmatrix} \quad 2.9$$

where r_m and θ_m are the measured range and bearing to the contact. θ_m is the angle measured from a vertical axes, or "True North" reference. The measurement covariance matrix must also be transformed from polar to cartesian coordinates. The resulting elements of the cartesian covariance matrix (Bar Shalom and Li, 1995):

$$\mathbf{R}_{\text{CART}} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

where

2.10

$$\begin{aligned} R_{11} &= r_m^2 \sigma_\theta^2 \sin^2 \theta_m + \sigma_r^2 \cos^2 \theta_m \\ R_{22} &= r_m^2 \sigma_\theta^2 \cos^2 \theta_m + \sigma_r^2 \sin^2 \theta_m \\ R_{12} &= (\sigma_r^2 - r_m^2 \sigma_\theta^2) \sin \theta_m \cos \theta_m \end{aligned}$$

where σ_r and σ_θ are the range and bearing standard deviations of the sensor. For the simulation, the measurement and covariance are transformed from polar to cartesian coordinates by the m-file **pole2cart**. The listing is contained in the Appendix.

Equipped with these models, a tracker can be implemented that will produce estimates from line of bearing or cartesian position reports. The measurement and sensor covariance can be determined either directly, as in the case of the line of bearing report, or by analyzing the error ellipse associated with a cartesian report. Since measurements are generated for the simulation in this thesis, a technique for linearizing range and bearing measurements to obtain the position in cartesian coordinates is also required. The development of a track splitting tracker in the upcoming chapter will draw on the models discussed here.

III. THEORETICAL BASIS FOR THE TRACK SPLITTING TRACKER

A. THE KALMAN FILTER

The Kalman Filter is a recursive algorithm for estimating the contact's current state vector from its past estimate and current noisy measurement. With accurately modeled plant dynamics and sensor noise, the algorithm minimizes the mean square error in the resulting estimates. Several excellent sources exist for detailed derivations of the Kalman Filter (Bar Shalom and Li, 1998). Only the resulting equations are presented here. The following definitions and notations will be used:

$\hat{\mathbf{x}}_{k+1|k}$ \equiv the estimate of state vector (\mathbf{x}) at time $k+1$ given the measurement at time k (prediction)

$\hat{\mathbf{x}}_{k|k}$ \equiv the estimate of state vector (\mathbf{x}) at time k given the measurement at time k (corrected estimate)

$\mathbf{P}_{k+1|k}$ \equiv the covariance of state vector (\mathbf{x}) at time $k+1$ given the measurement at time k

$\mathbf{P}_{k|k}$ \equiv the covariance of state vector (\mathbf{x}) at time k given the measurement at time k

The Kalman Filter recursion contains a prediction and a correction step. The mechanics of these steps are now described.

Prediction: The weighted least squares estimate and associated covariance matrix for the next sample time are predicted based on plant and noise models according to these equations:

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k \hat{\mathbf{x}}_{k|k} \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_k \mathbf{P}_{k|k} \mathbf{F}_k' + \mathbf{Q}_k\end{aligned}\tag{3.1}$$

Correction: The estimate and covariance matrix at time $k+1$ are corrected with the new (time $k+1$) measurement.

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1} \tilde{\mathbf{z}}_{k+1} \\ \mathbf{P}_{k+1|k+1} &= (\mathbf{I} - \mathbf{K}_{k+1} \mathbf{H}) \mathbf{P}_{k+1|k} (\mathbf{I} - \mathbf{K}_{k+1} \mathbf{H})' + \mathbf{K}_{k+1} \mathbf{R}_{k+1} \mathbf{K}_{k+1}'\end{aligned}\tag{3.2}$$

where

$$\begin{aligned}\tilde{\mathbf{z}}_{k+1} &= \mathbf{z}_{k+1} - \mathbf{H} \hat{\mathbf{x}}_{k+1|k} \\ \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k} \mathbf{H}' [\mathbf{H} \mathbf{P}_{k+1|k} \mathbf{H}' + \mathbf{R}_{k+1}]^{-1}\end{aligned}$$

The standard Kalman Filter presented above can be used when the plant and measurement models are linear, as in the case of cartesian position reports. The measurement matrix, \mathbf{H} , for the standard filter is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\tag{3.3}$$

The line of bearing report, however, requires a nonlinear measurement matrix since

$$\theta(x) = \arctan \left(\frac{x_1}{x_3} \right)\tag{3.4}$$

The Extended Kalman Filter provides a means to handle nonlinear state or measurement equations by linearizing the nonlinear equations about the estimate. For the line of bearing problem, the linearized measurement matrix obtained from a first order Taylor Series expansion is:

$$\mathbf{H}_k = \begin{bmatrix} \frac{x_1}{x_1^2 + x_3^2} & 0 & \frac{-x_3}{x_1^2 + x_3^2} & 0 \end{bmatrix}_{x=\hat{x}_{k+1|k}}\tag{3.5}$$

The measurement matrix is linearized about the predicted state and must be calculated with each new measurement. Once the linearized measurement matrix is obtained, it may be used in the correction step in the same manner that the static measurement matrix (Eqn. 3.3) is used with the standard Kalman Filter to obtain the corrected estimate.

Since the Kalman Filter is recursive, an initial state estimate and state covariance are required to begin the prediction and correction process. A first order polynomial curve fit method for obtaining these from two (cartesian) measurements may be used (Bar Shalom and Li, 1998). The latest measurement is used for the position estimate, while the velocity estimate is obtained from the difference in position. The initial covariance is calculated:

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ \Delta T & 0 & \Delta T & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{1}{\Delta T} & 0 & -\frac{1}{\Delta T} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{cart\ 2} & \emptyset \\ \emptyset & \mathbf{R}_{cart\ 1} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\Delta T} & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{\Delta T} \\ 0 & -\frac{1}{\Delta T} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\Delta T} \end{bmatrix} \quad 3.6$$

\mathbf{R}_{cart} is as derived in the previous chapter (Eqn. 2.10), with the subscript denoting the measurement with which the covariance matrix is associated. The m-file **kal2init** generates the filter initializations from the plant model and simulation parameters.

The Kalman Filter recursions form the backbone of the tracker algorithm, predicting and correcting the contact's state estimate based on the most recent state estimate, the plant model, and the most recent measurement of the contact. In the multitarget tracking problem, some data association decisions must be made to pair measurements with existing tracks before this process can begin. A technique for comparing the features of the measurement with the known features of the target can be employed to quickly reduce the

number of candidates for update. Also, a gating procedure will eliminate the most unlikely candidates by considering only the measurements within a region around the prediction.

B. MEASUREMENT REJECTION USING FEATURES

One technique employed in this study relies on the assumption that the features for both a contact being tracked and a false measurement are normally distributed with different statistics. The vector of features used here is:

$$\mathbf{y} \equiv \begin{bmatrix} \text{emitter frequency (GHz)} \\ \text{emitter pulse repetition frequency (pps)} \end{bmatrix} \quad 3.7$$

Clearly, not all sensors would include these features in the position report, so clutter rejection using features can not always be accomplished. The statistics of the feature random variables are denoted:

$$\begin{aligned} \bar{\mathbf{y}} &\equiv \begin{bmatrix} E[\text{frequency}] \\ E[\text{prf}] \end{bmatrix} \\ \Sigma &\equiv \begin{bmatrix} \sigma_{\text{frequency}} \\ \sigma_{\text{prf}} \end{bmatrix} \end{aligned} \quad 3.8$$

Distances are then formed to compare each measurement with statistics of the contact and clutter models. Here, the subscript "T" denotes target, and "C" denotes clutter, or false target:

$$\begin{aligned} d_T &\equiv (\mathbf{y} - \bar{\mathbf{y}}_T)' \Sigma_T^{-1} (\mathbf{y} - \bar{\mathbf{y}}_T) \\ d_C &\equiv (\mathbf{y} - \bar{\mathbf{y}}_C)' \Sigma_C^{-1} (\mathbf{y} - \bar{\mathbf{y}}_C) \end{aligned} \quad 3.9$$

Small distances indicate a measurement that closely matches the feature model, based on the feature statistics. Forming a ratio and establishing a threshold provides a criterion for rejecting a measurement as clutter:

$$\frac{d_r}{d_c} > \text{Threshold} \quad 3.10$$

Alternatively, when the clutter feature means are not known, a gating technique can be used with the target feature distance alone. In this case, measurements are rejected when d_T exceeds a Chi-Squared distribution threshold. Using a threshold of 6 will retain the correct measurement with 95% confidence while discarding measurements with features that conform poorly with the expected values. With both feature rejection techniques, measurements that meet the feature criteria remain valid candidates for update. Further processing will reduce the number of candidates based on the state estimate and position measurement.

C. ELLIPSOIDAL GATING

Just as a confidence ellipse exists around a noisy measurement, a region can be defined around an estimate that will establish the area within which a candidate observation must lie with some probability. The smallest region for a given probability is an ellipse defined by the innovation and innovation covariance resulting from a measurement, (Bar Shalom and Li, 1995):

$$[z - \hat{z}_{k+1|k}]^T \Sigma^{-1} [z - \hat{z}_{k+1|k}] \leq \gamma$$

where

3.11

$\Sigma = \mathbf{H} \mathbf{P}_{k|k-1} \mathbf{H}' + \mathbf{R} \equiv$ the innovation covariance

$\gamma \equiv$ chi-squared threshold

$z - \hat{z}_{k+1|k} \equiv$ the measurement's innovation

The above quadratic (norm of the innovation squared) is a Chi-Squared random variable with the number of degrees of freedom equal to the dimension of the measurement (two for cartesian, one for line of bearing). Thus, for each measurement received and each track held, the measurement innovation and innovation covariance must be calculated. With these, the norm of the innovation squared may be calculated and compared to the threshold. Norms exceeding the threshold indicate a pairing that is highly unlikely and need not be considered.

The application of this gating quadratic is straightforward for the cartesian position report. In these simulations, a threshold value of 6 has been used for cartesian reports, representing a 95% confidence gate. For line of bearing reports, the linearized measurement matrix must be used in determining the innovation covariance. The 95% threshold for this single degree of freedom distribution has been approximated as 4 in this simulation, but in reality is slightly less.

D. TRACK SPLITTING AND THE LIKELIHOOD FUNCTION

If only one measurement remains a candidate for track updating after processing with feature measurements and gates, no further tracker logic is required. The candidate

measurement is passed off to the Kalman Filter (standard or extended) and the corrected estimate can be obtained. If no measurements remain a candidate after processing, options can be included in the tracker to increase thresholds in the feature or ellipsoidal gate algorithms in an effort to obtain a candidate. (No such logic has been included in this simulation, however). When more than one measurement remains a candidate, the tracker is forced to make a decision as to which one most likely originated from the contact being tracked. With the track split approach, this decision is deferred until additional information (measurements) can be obtained. The predicted state is updated with each measurement (by the Kalman Filter correction step) so that a single track will become n tracks after processing, where n is the number of candidate measurements. For each of these tracks, a Kalman Filter prediction will be obtained so that a gate may be established for subsequent measurements. Feature processing, gating, splitting, and updating will be repeated for each track as new measurements are received.

With this logic, the number of tracks held in the system can grow exponentially. To aid in identifying the correct track, (that is, the one most closely matching the plant model) a likelihood function based on the Gaussian assumptions of the plant and measurement model is computed recursively each time a measurement updates a track (Bar Shalom and Li, 1998):

$$\lambda_k = \lambda_{k-1} + [\mathbf{z} - \hat{\mathbf{z}}_{k+1|k}]^T \Sigma^{-1} [\mathbf{z} - \hat{\mathbf{z}}_{k+1|k}] \quad 3.12$$

That is, the likelihood that a sequence of measurements originated from a track conforming to the model is the sum of the previous score and the norm of the innovation squared. A

low score indicates that the sequence conforms to the model well and is likely a valid track. Thus, a metric exists to aid in managing the number of tracks in the system

E. TRACK DROPPING CRITERIA

A Track Dropping algorithm is essential for a track splitting tracker to prevent overloading of the computational and data storage capacities of the system. Track scores, computed by the likelihood function (Eqn. 3.12), are one available tool for selecting the state estimates that have arisen out of the most unlikely measurement sequences. By choosing an appropriate threshold, the number of false tracks held in the system can be kept to a minimum without inadvertently dropping valid tracks. Another parameter that aids in track management is the most recent time at which an estimate was updated with a measurement, or update time. A threshold can be set for this parameter as well, triggering the tracker to automatically eliminate a held track once it is exceeded. A combination of these techniques is included in this simulation.

IV. COMPUTER SIMULATION

A. OVERVIEW

The listing for a simulation of a track splitting tracker written for MATLAB® is contained in the Appendix. This tracker incorporates the theoretical models previously discussed. Figure 1 shows the functional flow of the tracker simulation. Two dimensional constant velocity motion is simulated for two targets in a flat earth coordinate system. An instantaneous turn may be specified by selecting a time parameter for that occurrence. Two attributes, emitter frequency and pulse repetition frequency (PRF), may also be associated with the contacts. A range and bearing measurement of contact position from a fixed sensor position is made nearly every 90 minutes. The contact's features are also obtained at each measurement. Bearing-only measurement times may also be specified. Along with the noisy measurements of contact position, false measurements that did not originate from the contact are generated at sample times in an area around the contact's position. These false measurements have their own associated feature measurements. Thus, a position measurement and the associated feature measurements at a specific measurement time can be considered a single sensor report.

After generating all measurements at a sample time, track processing begins. The feature rejection loop compares the expected feature values for a given track with the measured values for each report. Reports that do not conform well to the track's expected features are removed from the set of candidates for the track being processed, but will be

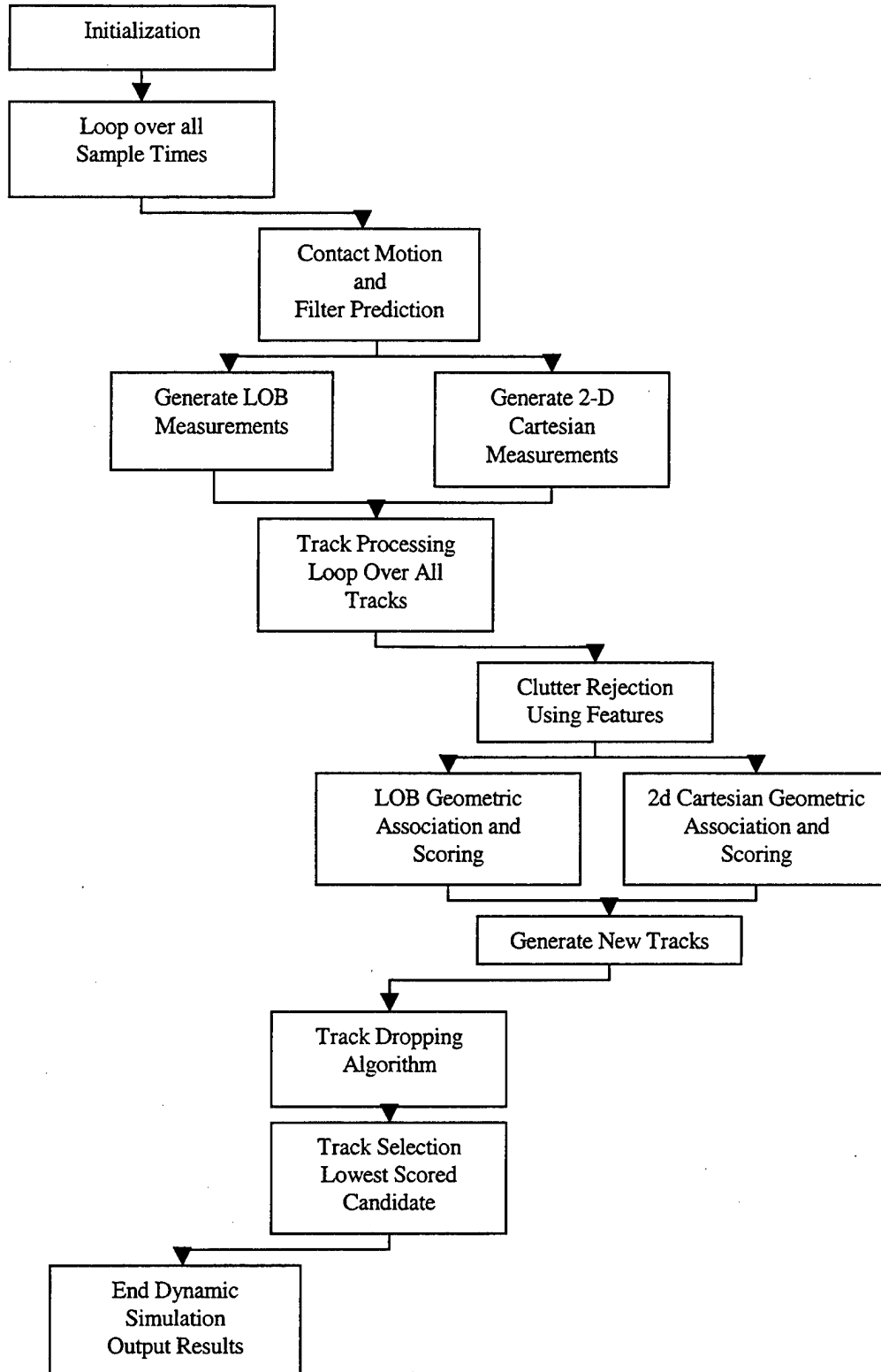


Figure 1. Functional Flow of Track-Splitting Simulation

considered for subsequent tracks. The remaining candidate reports are filtered through a geometric gate around the track's expected position. For all measurements that fall within the gate, a corrected state estimate and likelihood score are computed. If no measurement falls within the gate, the track is not updated at that sample time. Once all tracks have been processed with the measurements for a given sample time, a track management routine begins. This routine performs two functions: track dropping and track association. The track dropping function checks likelihood scores and estimate update times. If these exceed thresholds, the tracks are dropped and will not be considered in future processing. The track association function compares the scores of all estimates that are candidates for the original two targets, and selects the estimates that correspond with the lowest scores.

After looping through all measurement times, the simulation generates several output plots. True contact motion, noisy measurements, false measurements, predicted position estimates, and corrected position estimates are displayed in several figures. Also, a statistics matrix that is useful in tracker performance evaluation is displayed in the MATLAB command window. Items such as the number of tracks held, the number of clutter points generated, and the measurement and estimate error at each sample time can be reviewed.

B. SIMULATION 1: FEATURE REJECTION TEST

The first simulation demonstrates the ability of the feature rejection algorithm discussed in Section B of Chapter III to discriminate between false measurements and target

measurements. Here, clutter frequency and PRF means are well separated from the two targets relative to the standard deviation of the sensor's measurements. The parameters for the simulation are as follows:

Simulation:

number of measurements: 16

clutter density: 5 false targets per 60nm X 60nm area

Sensor:

Position: 18.00N 113.00W

σ_r : 5 nm

σ_θ : 0.5 degrees

σ_{freq} : 0.001 GHz

σ_{prf} : 0.1 pps

Mean Clutter Features:

$f_{clutter}$: 1.4 GHz

$prf_{clutter}$: 3.8 pps

Target 1:

Initial Position: 22.00N 110.00W

Heading: 145T

Speed: 13 Kts

Turns: To 180T at 590 minutes in simulation

f_{mean} : 1.406 GHz

prf_{mean} : 4.4 pps

q^2 : 1×10^{-8}

Target 2:

Initial Position: 22.00N 109.00W

Heading: 120T

Speed: 15 Kts

Turns: none

f_{mean} : 1.394 GHz

prf_{mean} : 3.2 pps

q^2 : 1×10^{-8}

Measurements: 14 cartesian measurements at about 90 minute intervals. Additionally, one

Line of Bearing measurement will be obtained 30 minutes before the 5th cartesian measurement. A second line of bearing measurement will be obtained 45 minutes before the 10th cartesian measurement.

Filter Settings and Thresholds:

Geographic gating: $\gamma = 6$, corresponding to 95% probability for a chi-squared distributed random variable.

Delete track score threshold: 48

Delete track time-late threshold: 200 minutes

Figure 2 shows the true track for targets one and two. Target one will follow the course change model, and target two will maintain a straight line track.

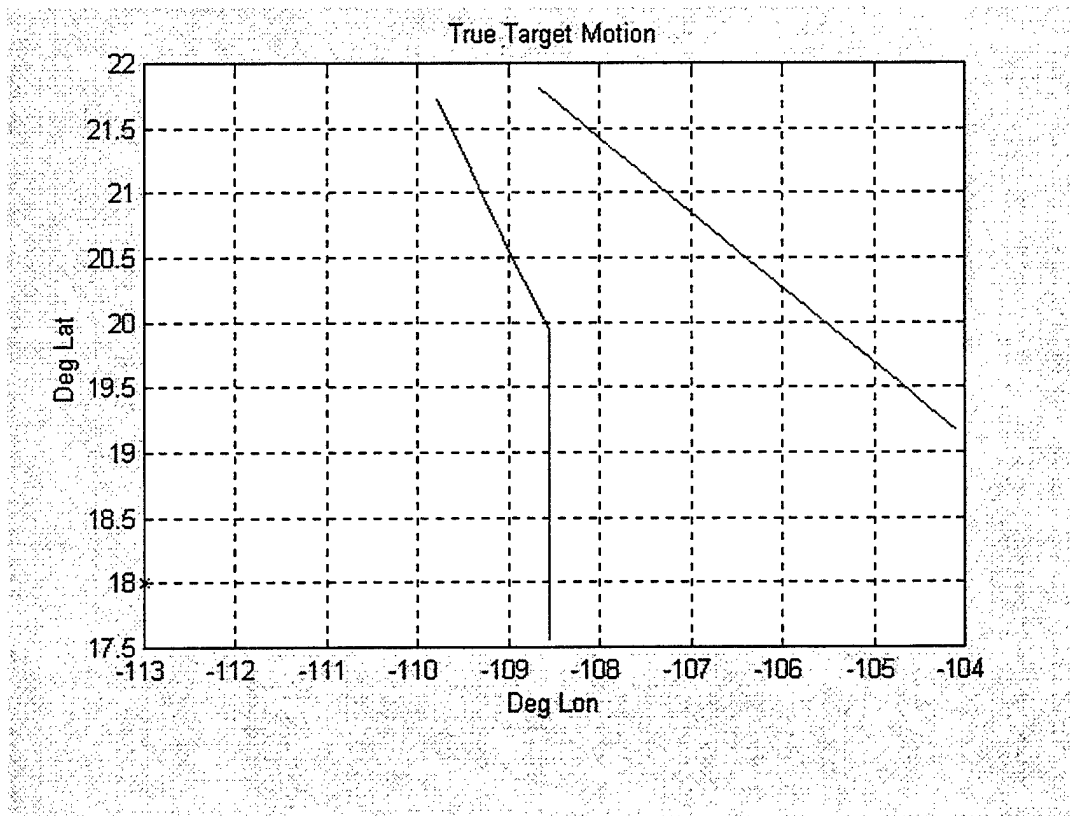


Figure 2. True Target Motion for Targets 1 and 2.

Figure 3 shows the true target position at sampling times, target and false measurements, and tracker position predictions ($\mathbf{z}_{k|k-1}$). The ellipses represent the 95% confidence regions around the predictions and are obtained from $\mathbf{P}_{k|k-1}$. These ellipses are related to the geographic gate used by the tracker. It is noted that many measurements are geographically feasible candidates, but it will be shown that, for this simulation, the feature rejection loop effectively removes most from consideration.

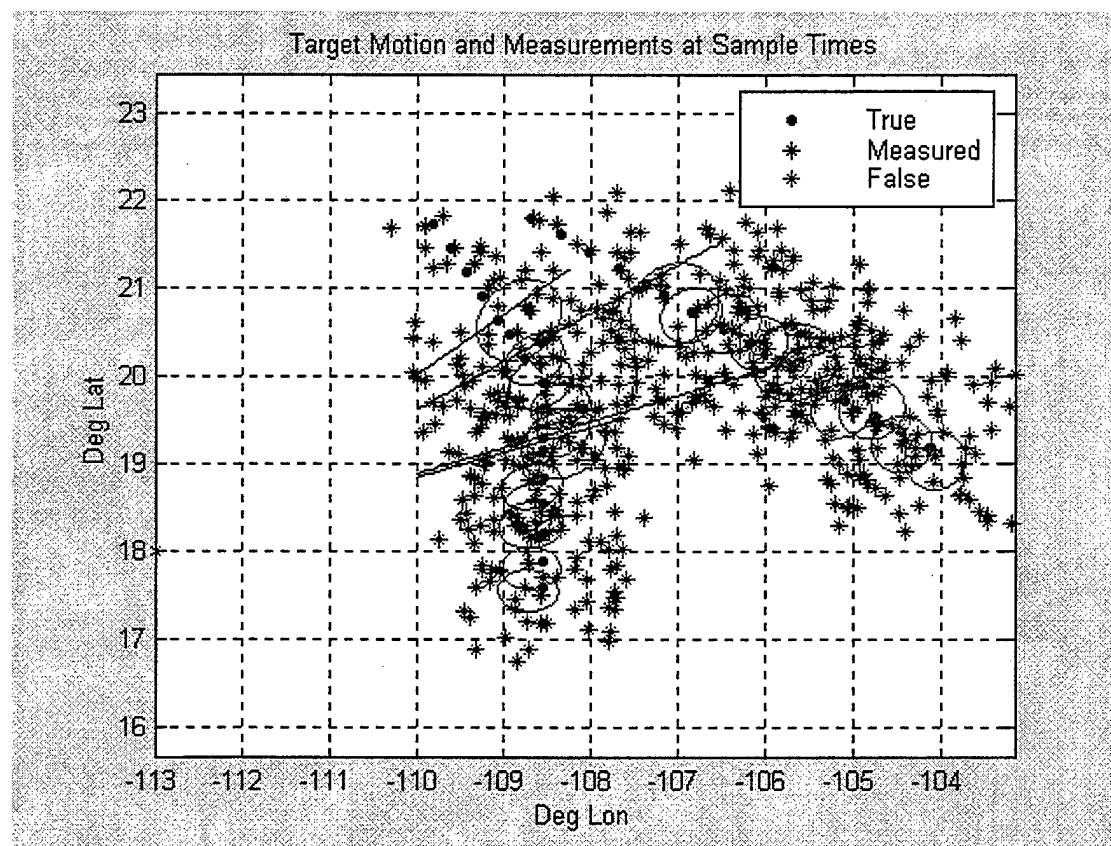


Figure 3. True Target Position and Measurements At Sample Times.
The ellipses are obtained from the prediction covariance and are related to the geographic gate.

Figure 4 shows true target position at sampling times, target measurements, and tracker corrected estimates ($\mathbf{z}_{k|k}$). In this figure, the error ellipses represent the 95%

confidence region around the corrected estimate and are obtained from $\mathbf{P}_{k|k}$. The longer Lines of Bearing correspond to target 2. The tracker is performing reasonably well with both targets. Estimate error increases during target 1's turn but quickly diminishes.

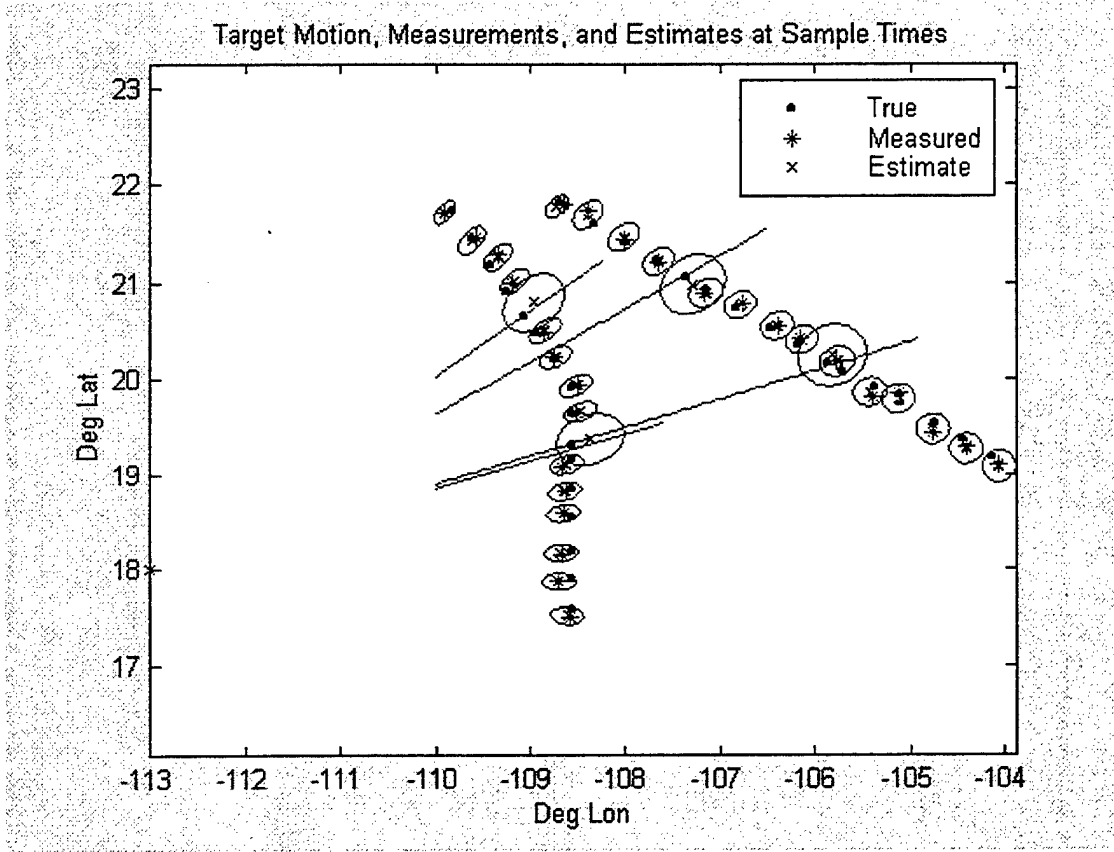


Figure 4. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.

Figure 5 contains the plot of Figure 4 for target 1 only. Likewise, Figure 6 is the target 2 portion of figure 4.

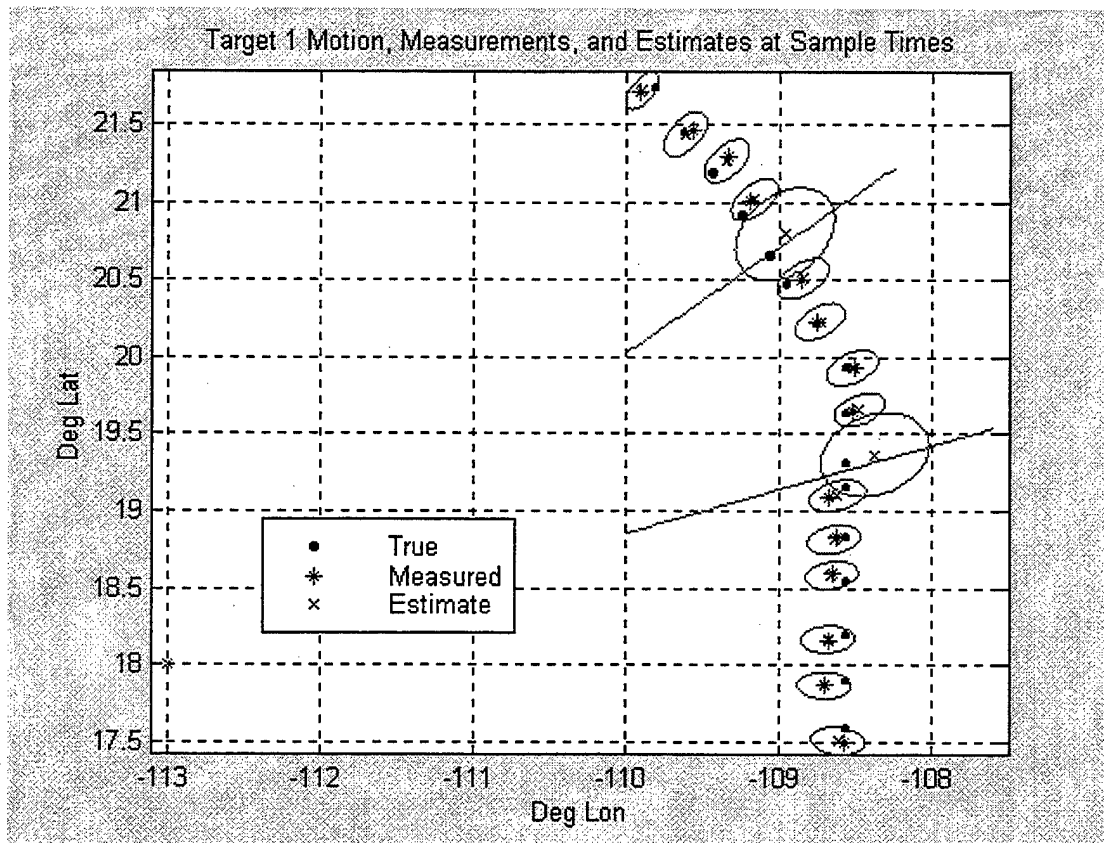


Figure 5. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times for Target 1.

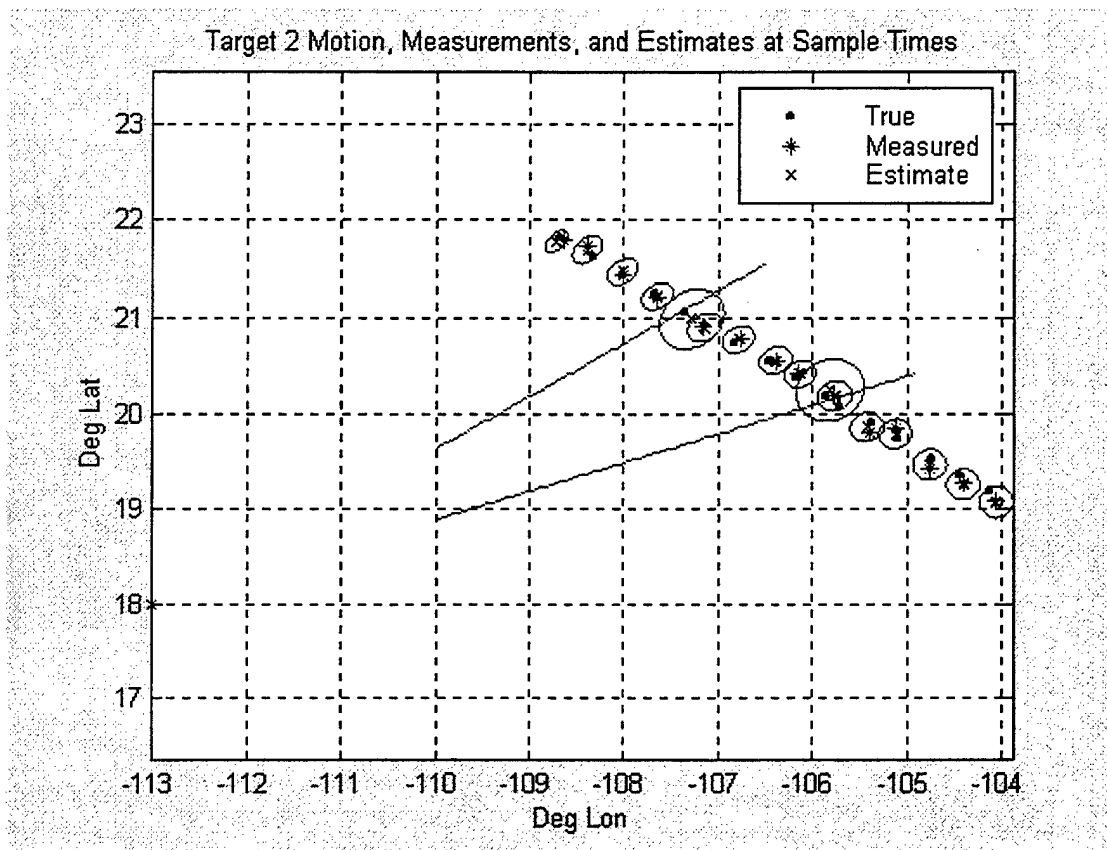


Figure 6. Simulation 1 True Target Position, Target Measurements, and Corrected Estimates at Sample Times for Target 2.

Table 1 is a compilation of statistics from the simulation. The following is an explanation of the fields:

index: measurement number

report type: "1" indicates a cartesian measurement sensor report (Lat/Long). "0" indicates a line of bearing measurement sensor report.

No of Clutter Msmts: The number of false measurements generated at that measurement time.

No of Valid Tracks: The number of tracks being processed by the tracker. It includes the original two targets plus new tracks that were generated by the track splitting algorithm, less tracks that were deleted for exceeding score or maximum time without update thresholds.

The following statistics are compiled for both targets:

No w/ Valid Attributes: The number of measurements at the sampling time that have emitter frequency and PRF parameters close enough to the target's expected values to pass the feature rejection loop.

No of Valid Measurements: The number of measurements at the sampling time to pass the feature rejection loop and fall within the range gate. If this number is "1", the track is updated with this measurement. If this number is greater than one, the measurement closest to the prediction will update the track, and remaining measurements will be used to generate new tracks (track split). If this number is "0", the track will not be updated at the sample time and output plots will contain the predicted, vice corrected, estimate.

Msmt error: Geometric distance in 60 nm (one degree) units between the target measurement and the true target position.

Estimate error: Geometric distance in 60 nm (one degree) units between the corrected estimate and the true target position.

Estimate swap: The track number of the candidate estimate with the lowest likelihood score. If this number is the same as the target number, the estimate for the target is the best available and no swapping will occur. If, however, this track number is different from the target number, the track numbers will be swapped so that the estimate for target one has the lowest likelihood score.

No of Trk Candidates: The number of tracks that are candidates for this target. These include the original track plus all tracks that were generated by valid measurements (those passing the feature rejection loop and falling within the gate), less those tracks that were deleted for exceeding score or maximum time without update thresholds.

Examination of the highlighted rows in Table 1 reveals the effectiveness of the feature rejection loop when target and clutter frequency and PRF means are well separated relative to the sensor's standard deviations. Despite a large number of measurements not originating from the target, few have frequency and PRF measurements close enough to the target expected values to be considered valid. In the case of the simulation results presented here, typically only one measurement successfully passes this loop per target, presumably originating from the target being processed.

index	1	2	3	4	5	6	7	8
report type	1	1	1	1	0	1	1	1
no of clutter msmts	0	0	0	0	0	82	42	48
no of valid tracks	2	2	2	2	2	2	2	2
target 1								
no of valid attributes	1	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1	1
msmt error	0.1065	0.0537	0.1439	0.1056	NaN	0.0908	0.0176	0.0490
estimate error	0.0900	0.0170	0.1175	0.1303	0.1725	0.0995	0.0302	0.0329
estimate swap	1	1	1	1	1	1	1	1
no of trk candidates	1	1	1	1	1	1	1	1
target 2								
no of valid attributes	1	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1	1
msmt error	0.0283	0.1311	0.0162	0.0448	NaN	0.0324	0.0693	0.0874
estimate error	0.0529	0.0977	0.0399	0.0385	0.1040	0.0339	0.0441	0.0817
estimate swap	2	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1	1
index	9	10	11	12	13	14	15	16
report type	1	0	1	1	1	1	1	1
no of clutter msmts	39	39	72	42	40	47	42	42
no of valid tracks	2	2	2	2	2	2	2	2
target 1								
no of valid attributes	1	1	3	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1	1
msmt error	0.0438	NaN	0.1330	0.0712	0.1047	0.1283	0.1503	0.0937
estimate error	0.0846	0.1930	0.0808	0.0869	0.1089	0.1281	0.1492	0.0995
estimate swap	1	1	1	1	1	1	1	1
no of trk candidates	1	1	1	1	1	1	1	1
target 2								
no of valid attributes	1	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1	1
msmt error	0.0539	NaN	0.1232	0.0918	0.1153	0.1139	0.1031	0.1292
estimate error	0.0635	0.0758	0.1120	0.0588	0.0645	0.0704	0.1090	0.1378
estimate swap	2	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1	1

Table 1. Statistics for Simulation 1: Feature Rejection Test

C. SIMULATION 2: GATING TEST

Another case to consider is when the two targets have similar frequency and PRF means. This will demonstrate the effectiveness of the geographic gating in selecting the correct candidate measurement. This simulation is conducted with the same parameters as the previous simulation, with the exception of the target feature means:

$$f_{mean}: 1.406 \text{ GHz}$$

$$prf_{mean}: 4.4 \text{ pps}$$

for both targets. The false measurement feature means remain unchanged, ensuring that the majority of these measurements will be rejected.

Figure 7 shows true target position at sampling times, target measurements, and tracker corrected estimates ($\mathbf{z}_{k|k}$). As with figure 4, the error ellipses represent the 95% confidence region around the corrected estimate and are obtained from $\mathbf{P}_{k|k}$. Plots of true target track and clutter measurements have been omitted since they are similar to Figures 2 and 3. As expected, the tracker is performing reasonably well with both targets and producing results that are similar to the case when features are well separated.

Analysis of the simulation results contained in Table 2 shows that, generally, the gating is effectively selecting the correct measurement. At each sample time, there are two measurements that have passed through the feature rejection loop, presumably corresponding to targets 1 and 2. Gating usually selects the correct measurement for use in updating the estimate, preventing a track split. Occasionally however, more than one

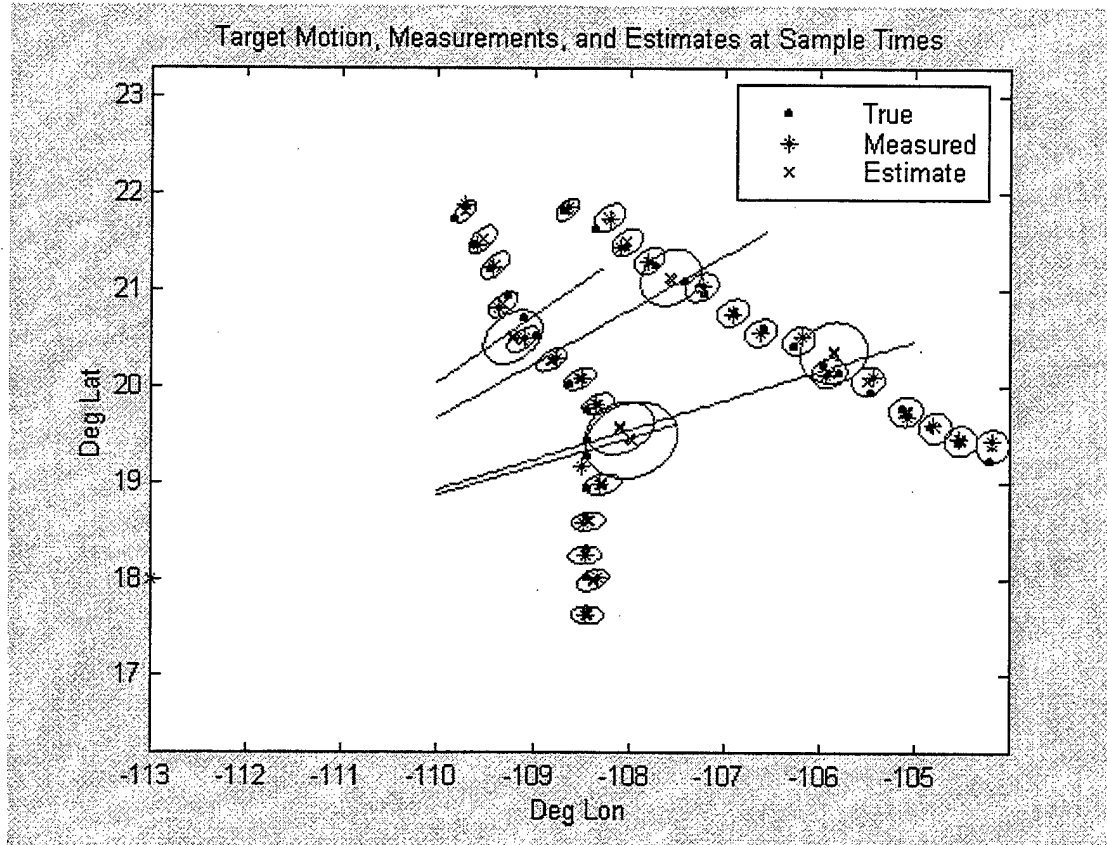


Figure 7. Simulation 2 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.

measurement is in the geographic gate and the track is split. These instances are limited to the Line of Bearing measurements (indices 5 and 10), and should be expected for the geometry of the simulation. Referring to Figure 3, the line of bearing measurements to both targets fall within the ellipses for both predicted estimates, indicating a track split situation. The track is split by updating the prediction with both line of bearing measurements, one for each track. The generated tracks have state estimates that are very close to the updated track, making them nearly indistinguishable in figure 7.

index	1	2	3	4	5	6	7	8
report type	1	1	1	1	0	1	1	1
no of clutter msmts	0	0	0	0	0	74	41	41
no of valid tracks	2	2	2	2	4	4	4	4
target 1								
no of valid attributes	2	2	2	2	2	2	2	2
no of valid msmts	1	1	1	1	2	1	1	1
msmt error	0.1584	0.0286	0.0336	0.1654	NaN	0.1298	0.0353	0.1480
estimate error	0.1174	0.1008	0.0581	0.1168	0.2270	0.1458	0.0280	0.1222
estimate swap	1	1	1	1	1	1	1	1
no of trk candidates	1	1	1	1	2	2	2	2
target 2								
no of valid attributes	2	2	2	2	2	2	2	2
no of valid msmts	1	1	1	1	2	1	1	1
msmt error	0.0638	0.1803	0.0583	0.1018	NaN	0.0745	0.0354	0.0529
estimate error	0.0341	0.1815	0.0528	0.0908	0.1547	0.0751	0.0189	0.0485
estimate swap	2	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	2	2	2	2
index	9	10	11	12	13	14	15	16
report type	1	0	1	1	1	1	1	1
no of clutter msmts	42	42	74	115	45	43	40	47
no of valid tracks	4	8	8	8	8	8	8	8
target 1								
no of valid attributes	2	2	2	2	2	2	2	2
no of valid msmts	1	2	0	1	1	1	1	1
msmt error	0.0936	NaN	0.1133	0.1279	0.0443	0.0564	0.1033	0.0692
estimate error	0.1283	0.3668	0.4907	0.1643	0.0224	0.0620	0.0620	0.0634
estimate swap	1	1	1	1	1	1	5	1
no of trk candidates	2	4	4	4	4	4	4	4
target 2								
no of valid attributes	2	2	2	2	2	2	2	2
no of valid msmts	1	2	1	1	1	1	1	1
msmt error	0.1269	NaN	0.1548	0.1568	0.0887	0.0255	0.0426	0.2025
estimate error	0.0793	0.1603	0.1153	0.1263	0.0411	0.0387	0.0301	0.1650
estimate swap	2	2	2	2	2	2	2	2
no of trk candidates	2	4	4	4	4	4	4	4

Table 2. Statistics for Simulation 2: Gating Test

D. SIMULATION 3: SCORING TEST

The tracker's ability to select the best track candidate is examined in a simulation where the clutter's feature means are relatively close (within 3 standard deviations) of a target's means:

Mean Clutter Features:

$$f_{clutter}: 1.4 \text{ GHz}$$

$$prf_{clutter}: 3.8 \text{ pps}$$

Mean Target 1 Features:

$$f_{mean}: 1.403 \text{ GHz}$$

$$prf_{mean}: 4.1 \text{ pps}$$

The tracker will have several false measurements successfully pass the feature rejection loop for target 1. Of these measurements, it is expected that some will fall within the geographic gate causing the track to be split. With successive measurements, track scoring is used to select the estimate that most closely conforms to the motion model and eliminate unlikely tracks. The number of measurements for this simulation is increased from 16 to 20 to better show the track management capabilities. Mean features for target 2 are as specified in Simulation 1 and are expected to produce similar results.

Figure 8 shows true target position at sampling times, target measurements, and tracker corrected estimates ($\mathbf{z}_{k|k}$). All tracks are shown here, including those generated as a result of splitting the original tracks. However, error ellipses are plotted only around the

estimates most likely to be targets 1 and 2 (i.e., those two candidate tracks with the lowest track scores). It is apparent that several new tracks have been generated for target 1, and that the track with the lowest score conforms to the true target track well. Figure 9 shows true target position at sampling times, target measurements, and tracker corrected estimates ($\mathbf{z}_{k|k}$) for the most likely track estimate only, illustrating this further.

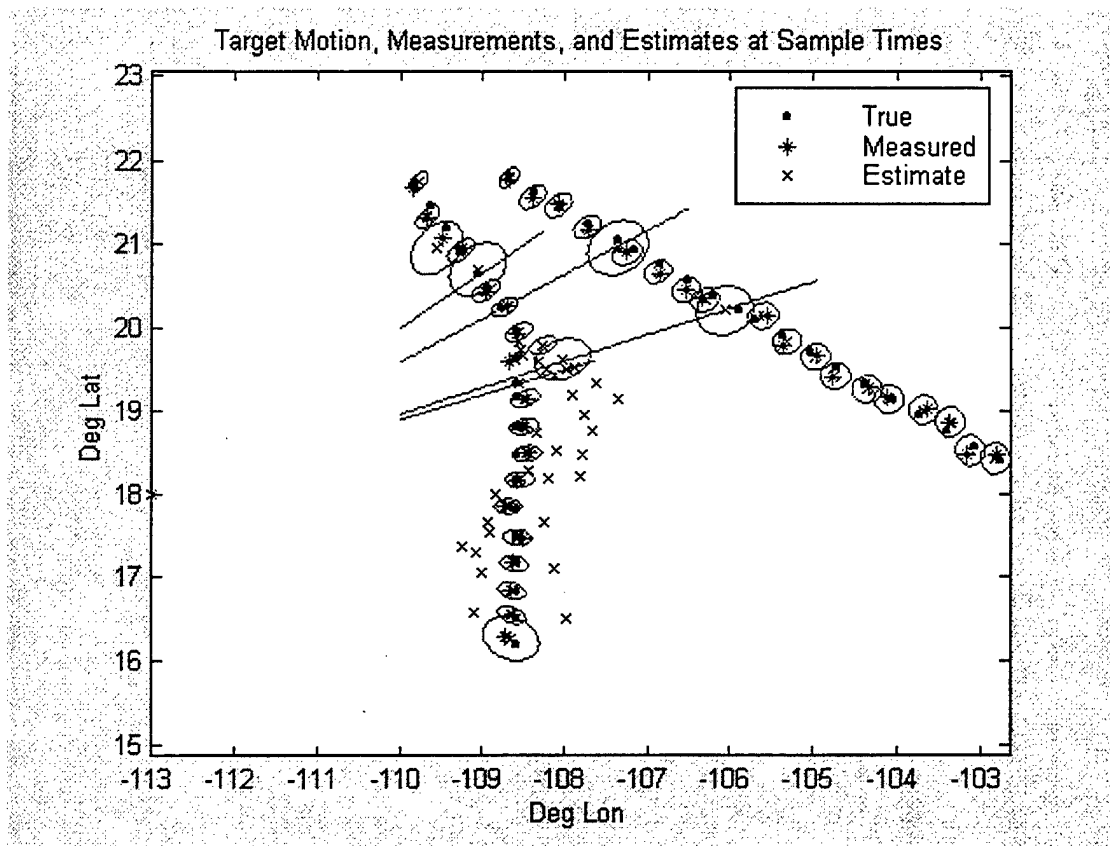


Figure 8. Simulation 3 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.

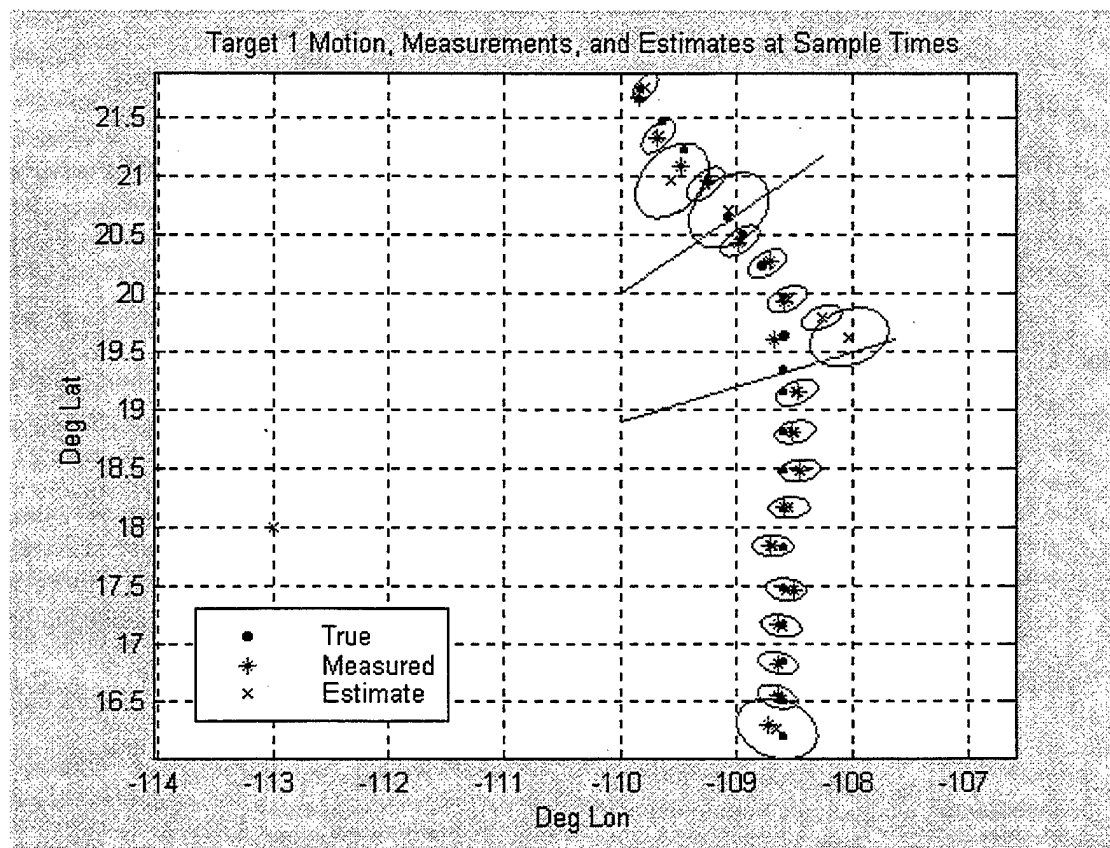


Figure 9. Simulation 3 True Target Position, Target Measurements, and Corrected Estimates at Sample Times. Best Estimate Only For Target 1.

An understanding of the track management capability for this tracker can be obtained by examining Table 3. Multiple measurements possess valid features and pass through the feature rejection loop at many measurement times. Also, track splitting occurs when more than one measurement falls within the geographic gate: samples 9 and 12. These new tracks add to the list of track candidates for target 1. The number of track candidates increases from 1 to 3 at sample 9 (the first split), then peaks at 7 candidates at sample 13. It recedes to 2 candidates by the end of the simulation as a result of track dropping. By comparing the track score for all track candidates, the tracker identifies estimates that are

more likely to be target 1 than the currently held estimate on one occasion. At index 11, track 3 is selected and becomes the new track 1. The track shown in Figure 7 is the sequence of estimates with the lowest track score at each sample time.

index	1	2	3	4	5	6	7
report type	1	1	1	1	0	1	1
no of clutter msmts	0	0	0	0	0	86	39
no of valid tracks	2	2	2	2	2	2	2
target 1							
no of valid attributes	1	1	0	1	1	7	3
no of valid msmts	1	1	0	1	1	1	1
msmt error	0.0698	0.1495	0.1311	0.0244	NaN	0.0640	0.0719
estimate error	0.0366	0.1360	0.2751	0.0138	0.0502	0.0518	0.0440
estimate swap	1	1	1	1	1	1	1
no of trk candidates	1	1	1	1	1	1	1
target 2							
no of valid attributes	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1
msmt error	0.0448	0.0685	0.0574	0.0721	NaN	0.1022	0.1017
estimate error	0.0183	0.0648	0.0446	0.0500	0.0938	0.0941	0.0934
estimate swap	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1

Table 3. Statistics for Simulation 3: Scoring Test

index	8	9	10	11	12	13	14
report type	1	1	0	1	1	1	1
no of clutter msmts	45	43	43	72	48	46	45
no of valid tracks	2	4	4	5	7	8	8

target 1

no of valid attributes	3	5	5	7	5	5	7
no of valid msmts	1	3	1	0	2	1	1
msmt error	0.0304	0.0973	NaN	0.1141	0.0611	0.1346	0.0100
estimate error	0.0269	0.3519	0.6189	0.1039	0.0890	0.1273	0.0374
estimate swap	1	1	1	3	1	1	1
no of trk candidates	1	3	3	4	6	7	7

target 2

no of valid attributes	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1
msmt error	0.1084	0.1509	NaN	0.1714	0.1044	0.0727	0.1237
estimate error	0.1122	0.1233	0.1747	0.1177	0.0805	0.0818	0.1055
estimate swap	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1

index	15	16	17	18	19	20
report type	1	1	1	1	1	1
no of clutter msmts	46	46	46	47	50	46
no of valid tracks	8	7	4	3	3	3

target 1

no of valid attributes	2	8	2	8	2	2
no of valid msmts	1	1	1	1	1	0
msmt error	0.1348	0.0871	0.0535	0.0624	0.0730	0.1711
estimate error	0.1079	0.0205	0.0287	0.0572	0.0716	0.1013
estimate swap	1	1	1	1	1	1
no of trk candidates	7	6	3	2	2	2

target 2

no of valid attributes	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1
msmt error	0.0684	0.0740	0.0996	0.0962	0.1436	0.1052
estimate error	0.0729	0.0490	0.0667	0.1037	0.0758	0.0755
estimate swap	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1

Table 3 (Continued). Statistics for Simulation 3: Scoring Test

Repeated simulations with these initial parameters produce results that vary in detail but are generally the same in overall performance. The tracker consistently obtains a sequence of estimates that conforms with the true track reasonably well. Differences between simulations are found in the number and occurrence of track splitting. Also, some simulations have more estimate swaps than presented here, while others have less. This is expected due to the randomness of the clutter and target measurements. The track scoring and comparisons performed by the tracker have demonstrated considerable strength in separating sequences that arise from uniformly distributed clutter measurements from those that arise from Gaussian distributed target measurements.

E. SIMULATION 4: COMBINED TEST

Finally, the case is considered where false target features are broadly distributed over a range compared to the target features. This is the most realistic of the scenarios presented, since it models target tracking in an environment where many other contacts are operating with various emitters. A combination of the tracker's capabilities will be required to maintain a reasonable track: feature rejection, gating, and track scoring. Furthermore, it will be assumed that the tracker does not know the feature mean for the false measurements and the alternate feature rejection technique discussed in Section B of Chapter III will be used. For the purpose of generating false target feature measurements, a normal distribution is used with the following statistics:

Mean Clutter Features:

$f_{clutter}$: 1.4 GHz

$prf_{clutter}$: 3.8 pps

Clutter Standard Deviation:

σ_{freq} : 1.000 GHz

σ_{prf} : 3.0 pps

Target features are normally distributed as specified in Simulation 1:

Target 1:

f_{mean} : 1.406 GHz

prf_{mean} : 4.4 pps

Target 2:

f_{mean} : 1.394 GHz

prf_{mean} : 3.8 pps

Target Standard Deviation:

σ_{freq} : 0.001 GHz

σ_{prf} : 0.1 pps

Figure 10 shows true target position at sampling times, target measurements, and tracker corrected estimates ($\mathbf{z}_{k|k}$). As before, all tracks are shown here with error ellipses plotted only around the estimates most likely to be targets 1 and 2. The tracker is performing reasonably well with both targets. Some new tracks have been generated around target 1, but the most likely state estimate conforms well with true target position.

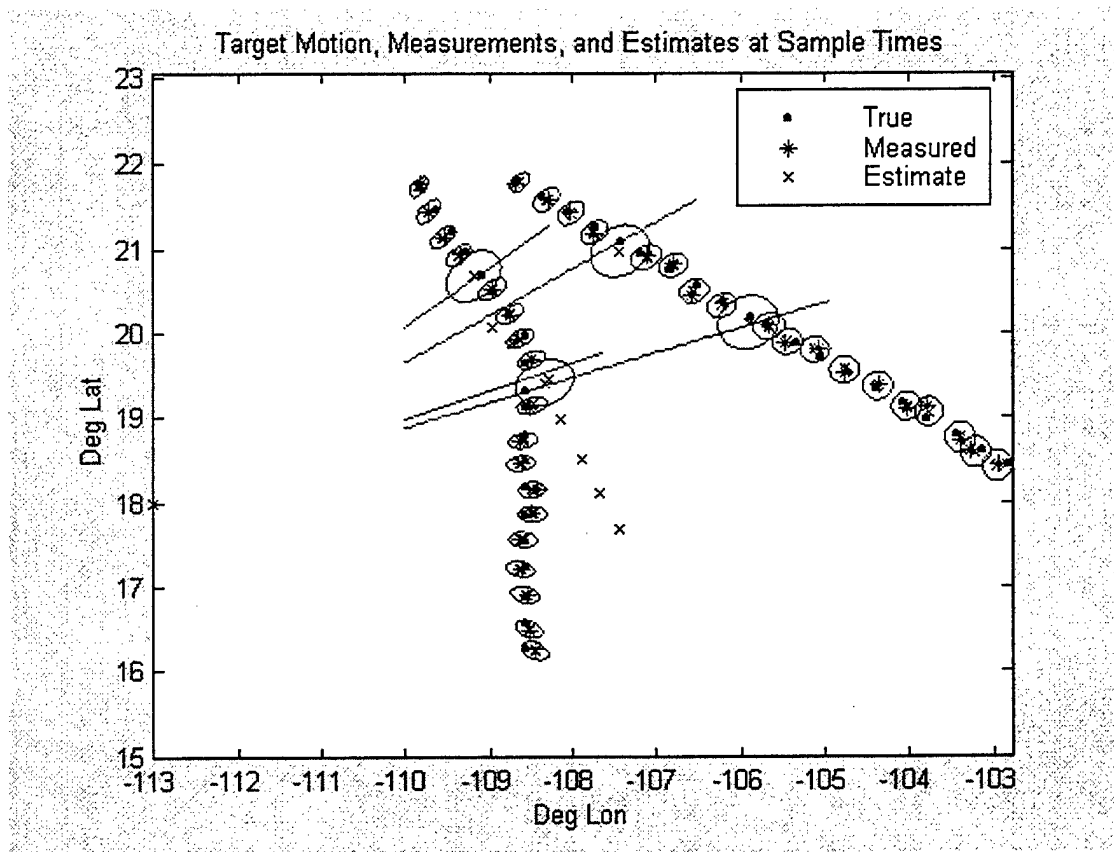


Figure 10. Simulation 4 True Target Position, Target Measurements, and Corrected Estimates at Sample Times.

Table 4 shows the output statistics for this simulation. It is noted that a few false measurements pass through the feature rejection loop frequently, but track splitting occurs rarely. There is only one track split apparent with target 1, and that occurs at sample 7. Target 2 does not exhibit any track splitting. The number of valid tracks increases from 2 to 3 at sample 7, then from 3 to 4 at sample 11. Since there is only one valid measurement in the gate for both targets at this time, it suggests that the track generated at sample 7 was split. The number of valid tracks recedes to 3 at index 15, where it remains until the end of the simulation. It is also noted that none of the generated tracks ever achieve a lower score than the original two tracks and thus no estimate swapping occurs.

index	1	2	3	4	5	6	7
report type	1	1	1	1	0	1	1
no of clutter msmts	0	0	0	0	0	87	44
no of valid tracks	2	2	2	2	2	2	3
target 1							
no of valid attributes	1	1	1	1	1	2	2
no of valid msmts	1	1	1	1	1	1	2
msmt error	0.0355	0.1014	0.1158	0.0805	NaN	0.0263	0.0098
estimate error	0.0323	0.0898	0.1222	0.0978	0.1060	0.0288	0.0019
estimate swap	1	1	1	1	1	1	1
no of trk candidates	1	1	1	1	1	1	2
target 2							
no of valid attributes	1	1	1	1	1	3	2
no of valid msmts	1	1	1	1	1	1	1
msmt error	0.0421	0.0833	0.0262	0.0819	NaN	0.0804	0.0829
estimate error	0.0374	0.0695	0.0372	0.0663	0.1019	0.0723	0.0725
estimate swap	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1

Table 4. Statistics for Simulation 4: Combined Test

index	8	9	10	11	12	13	14
report type	1	1	0	1	1	1	1
no of clutter msmts	42	42	42	81	48	40	43
no of valid tracks	3	3	3	4	4	4	4

target 1

no of valid attributes	2	2	2	1	2	2	1
no of valid msmts	1	1	1	1	1	1	1
msmt error	0.0929	0.0953	NaN	0.0496	0.0846	0.0490	0.1498
estimate error	0.0636	0.0945	0.2575	0.0888	0.0564	0.0596	0.0887
estimate swap	1	1	1	1	1	1	1
no of trk candidates	2	2	2	3	3	3	3

target 2

no of valid attributes	1	1	1	1	1	1	1
no of valid msmts	1	1	1	1	1	1	1
msmt error	0.1319	0.0371	NaN	0.0363	0.1444	0.1112	0.0291
estimate error	0.0753	0.0623	0.0786	0.0276	0.1021	0.1122	0.0518
estimate swap	2	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1	1

index	15	16	17	18	19	20
report type	1	1	1	1	1	1
no of clutter msmts	46	44	42	50	45	45
no of valid tracks	3	3	3	3	3	3

target 1

no of valid attributes	1	1	1	2	2	1
no of valid msmts	1	1	1	1	1	1
msmt error	0.0775	0.0756	0.0542	0.0384	0.1297	0.1292
estimate error	0.0974	0.0391	0.0600	0.0176	0.0966	0.1287
estimate swap	1	1	1	1	1	1
no of trk candidates	2	2	2	2	2	2

target 2

no of valid attributes	1	1	1	2	2	1
no of valid msmts	1	1	1	1	1	1
msmt error	0.0436	0.0706	0.1265	0.0961	0.1294	0.1203
estimate error	0.0329	0.0600	0.0749	0.0536	0.0859	0.1296
estimate swap	2	2	2	2	2	2
no of trk candidates	1	1	1	1	1	1

Table 4 (Continued). Statistics for Simulation 4: Combined Test

Just as with the previous simulations, repeated runs with these initial parameters produce results that vary in detail but are consistent in overall performance. Several runs were observed with no occurrence of track splitting. On the other hand, many simulations produced satisfactory results, but with track splitting and estimate swapping. The tracker's proven capabilities in splitting, scoring, and swapping result in target estimates that conform reasonably well with truth.

F. SIMULATIONS WITH EXTERNAL MEASUREMENT DATA

Much of the code contained in the Appendix is devoted to generating data that the tracker would receive from the sensor report. Further testing of the algorithm could be accomplished by stripping off this code and reading in collected sensor reports in the form of an external data file. With minor modifications, the tracker could be tested with position reports on two real tracks and several false measurements. Sensor reports must contain the following information:

Time of Measurement

Measurement Type: a flag for indicating Cartesian (Latitude and Longitude) or

Line of Bearing processing.

Position Measurement

Error Ellipse: The sensor covariance can be derived from this as discussed in

Chapter II.

Emitter Frequency

Frequency Measurement Covariance

Emitter PRF

Emitter PRF Measurement Covariance

Because the computer simulation in the Appendix is designed to evaluate the track maintenance capability of a track splitting approach, care must be taken in initializing the tracker. In the Appendix, the state estimates for Targets 1 and 2 are initialized by passing two noisy measurements of target position to the supporting m-file **kal2init**. The initial state estimate and covariance are then obtained as described in Chapter III. A similar approach could be taken when using external data by reading in two measurements for each target and passing them to **kal2init**. Another approach would be to simply provide the initial estimate as *a priori* knowledge and proceed directly to the main loop in the code. Similarly, emitter frequency and PRF means are provided as *a priori* knowledge in the Appendix code. When testing with external data, these *a priori* estimates must be consistent with the data for satisfactory performance.

V. CONCLUSION

A. SUMMARY

This research has developed a simulation to explore the use of a track splitting tracker in the maritime surveillance problem. The tracker uses Kalman Filtering to estimate target state from received measurements, which may be latitude and longitude, or line of bearing reports. Prior to filter update, measurements are filtered on the basis of features and geographic gates to eliminate unlikely measurement candidates. When more than one measurement falls within a track's gate, the track is split to allow subsequent measurements to help determine validity. Track scoring with a likelihood function serves as the basis for determining which track estimate originated from the modeled target's sequence of measurements. To prevent saturation of the tracker with false tracks, a track dropping routine eliminates the least likely estimates by considering track scores and time since last update.

The effectiveness of measurement rejection on the basis of features was demonstrated in simulation by assigning feature means to the false measurements that were well separated from the true target means. Examination of the simulation results showed that few or no clutter measurements passed through this rejection loop, while the measurements originating from the target generally did. The Kalman Filter track estimates conformed well with the true target tracks. To investigate the effectiveness of the geographic gating, both targets were assigned the same feature means. Since the targets

were well separated geographically but had similar features, both target measurements successfully passed the feature rejection test and were then correctly paired with a track on the basis of the geographic gate. Finally, the ability of the tracker to select the best track estimate from the set of candidate tracks was tested by assigning similar feature means to the false measurements and one of the targets. During the simulation, the track was split several times as a result of false measurements passing both the feature rejection and geographic gate tests. The tracker produced an estimate of target track that conformed well with truth by selecting the lowest scored estimate at each measurement time. These simulations demonstrated several desirable features for a maritime surveillance tracker.

B. FURTHER RESEARCH

With minor modifications, the code in the Appendix can be tested with a real data set. Initially, testing should resemble the simulations presented here with two real targets and several false measurements at each measurement time. Target feature means must be known before hand, and target tracks must be initialized before clutter measurements are introduced. Such a test would confirm the viability of the track splitting tracker in a simple, real-world application.

The algorithm can be significantly enhanced by adding a more sophisticated track initialization routine. Presently, the simulation tests the ability of the track splitting approach to maintain target tracks once obtained. Measurements that do not pass the feature test and fall within the geographic gate are discarded. An initialization routine

might establish tentative tracks with these measurements, which could then become regular tracks when scoring or number of update thresholds are met.

A more accurate geographic model can be developed for this tracker. Here, distances are converted to geographic latitude and longitude with a flat earth model. Since sensors may operate over great distances in maritime surveillance, a spherical earth model should be used.

APPENDIX

This Appendix contains the track-splitting tracker simulation and supporting code.

It was implemented in MATLAB® 5.1.

Main Simulation Code: trackSplit.m

%trackSplit.m

% This file simulates a multi-target, multi-sensor, track splitting tracker. Two
%dimensional motion is simulated using a constant velocity discrete time target motion
%model for two contacts (Target 1 and Target 2). An instantaneous turn may be
%specified by selecting a time to turn and entering that value for "turnTime" below.
%Targets also have emitter frequency and prf attributes.
% A measurement of target position and attributes are obtained "about every 90 min".
%Both targets are measured at the same instant. Measurement noise is modeled as a zero
%mean Gaussian Random Variable with variance as set in the SENSOR part of the code.
%The position measurements may be taken as Range and Bearing or Bearing only.
%Measurements are taken as Range and Bearing except on measurement indices that
%have been specified for Bearing only measurements. Also, an IF statement is in place
%for use in "skipping" measurements (missed measurements).
% Clutter measurements are also generated with target range and bearing
%measurements. Clutter measurements are modeled as uniformly distributed within a
%region around true track position. These measurements are assumed to have frequency
%and prf attributes that are also modeled as Gaussian random variables. These statistics
%are specified in the SENSOR part of the code. The clutter region is related to the
%eigenvalues of the estimate covariance matrix, $P(k|k-1)$, which may extremely large for
%tracks in their infancy. For this reason, an IF statement controls when clutter
%measurements begin.
% Clutter rejection is accomplished by comparing attributes of all measurements with
%the track's expected attributes. A weighted distance between clutter and target attribute
%means is calculated for each measurement and the ratio is formed. Measurements are
%rejected as clutter when the ratio exceeds a threshold, "attThresh".
% From the remaining measurements, Geometric Association is accomplished by a
%comparison is made by comparing expected target position ($z(k|k-1)$) and measurement
%position. The measurement that produces a track with the lowest log-likelihood score
%is selected for filter update. New Tracks are generated for all remaining measurements
%in the validation region (Chi-Square Distribution 95% Confidence Ellipse). A kalman
%filter performs a correction step using the selected measurement, resulting in $z(k|k)$. An
%error ellipse is obtained from the covariance of error matrix for plotting.

```

% Track Dropping logic is included to control the number of tracks in the system.
%After all tracks have been processed at a given measurement time, track scores and
%last update time are compared to threshold values (maxScore and maxTime). Tracks
%that exceed the threshold are dropped.
% Output plots are:
%      True target motion
%      True target motion and measurements
%      True target motion, measurements, and corrected estimates (x(k|k)) with
%      error ellipse
%      True target motion, measurements, and corrected estimates for target 1
%      True target motion, measurements, and corrected estimates for target 2

% This simulation requires the following additional m-files:
%      kal2init.m      {initializes the filter from 2 measurements}
%      elipa.m         {obtains error ellipse from P matrix}
%      pole2cart.m     {converts range and bearing measurements to
%      "               cartesian coords}
%
%      get2d.m         {generates cartesian position reports
%      "               (measurements)}
%
%      getLob.m        {generates LOB position reports}
%      getClut.m       {generates "clutter" cartesian position reports}
%      getClutFeature.m {generates "clutter" features reports}

clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Simulation and Sensor Initialization%%%%%%%%
%Define number of measurements to be taken:
nmeas = 16;           %number of measurements
clutDens = 5;         %defines clutter density: 5 tgt/3600 sqr nm

%sensor
slp = [-113.00;18.00]; %sensor location in lon;lat deg
sigma1r = 5/60;        %2.5 nm of uncertainty in range
sigma1b = .5*pi/180;   %1/2 degrees uncertainty in brg

slv = diag([sigma1r^2;sigma1b^2]); %covariance matrix for uncorrelated
%range and brg msmts
sigmaF = 1e-3;         %100 Hz uncertainty in f
sigmaPrf = 0.1;        %0.1 pps uncertainty in prf

fClut = 1.400;         %clutter feature statistical means
prfClut = 3.8;         %Ghz and pps

```

```

sigmaFClut = 1; %clutter feature statistical STD
sigmaPrfClut = 3;
sigC = diag([1/sigmaFClut; 1/sigmaPrfClut]); %parameter "packaging" for use in
sigT = diag([1/sigmaF; 1/sigmaPrf]); %clutter rejection loop
attClut = [fClut;prfClut];
c = sqrt(6); %error ellipsoids for 95% confidence
gamma = 6; %parameter for 95% threshold of chi squared dist
attThresh = 10; %threshold for attribute discrimination

maxScore = 48; %Track dropping parameters
maxTime = 270;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Target Initialization%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialize the Target 1 state vector of positions and velocities
hdg = 145*pi/180; %Target heading
speed = 13/3600; %Target speed, kts
xi = [-110.00; %Initial x posit (deg lon)
      speed*sin(hdg); %Initial x speed (y/s)
      22.00; %Initial y posit (deg lat)
      speed*cos(hdg)]; %Initial y speed (y/s)

turnTime = 590; %turn 590 minutes into simulation

%Attribute Data for Target 1
fTrue = 1.406; %true emitter Freq (GHz)
prfTrue = 4.4; %true emitter prf (pps)
attTrack(:,1) = [fTrue;prfTrue]; %"packaging"

%Initialize the Target 2 state vector of positions and velocities
hdg2 = 120*pi/180; %Target heading
speed2 = 15/3600; %Target speed, kts
xi2 = [-109.00; %Initial x posit (deg lon)
      speed2*sin(hdg2); %Initial x speed (y/s)
      22.00; %Initial y posit (deg lat)
      speed2*cos(hdg2)]; %Initial y speed (y/s)

turnTime2 = 1e6; %no turn for tgt 2

%Attribute Data for Target 2
fTrue2 = 1.394; %true emitter Freq (GHz)
prfTrue2 = 3.2; %true emitter prf (pps)
attTrack(:,2) = [fTrue2;prfTrue2]; %"packaging"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Tracker Parameter Initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%Define the H (measurement) matrix for cartesian measurement
H = [1 0 0 0;
      0 0 1 0];

%Define Target 1 covariance of plant noise matrix, Q
qsqr = 1e-8; %adjustable to account for
              %plant model deviations
del = 90; %mean sample period

Q = qsqr*[del^3/3 del^2/2 0 0;
          del^2/2 del 0 0;
          0 0 del^3/3 del^2/2;
          0 0 del^2/2 del];

%Define Target 2 covariance of plant noise matrix, Q
qsqr2 = 1e-8;
del = 90;

Q2 = qsqr2*[del^3/3 del^2/2 0 0;
             del^2/2 del 0 0;
             0 0 del^3/3 del^2/2;
             0 0 del^2/2 del];

%Build the measurement time vector (msmts taken "about" every 90 min)
rbMeas = 90*ones(nmeas-3,1) + 4*randn(nmeas-3,1); %range/brg measmt
                                                    %intervals in minutes

lob1 = rbMeas(5)-30; %get a line of brg msmt 30 min
                    %before 4th r/b msmt
lob2 = rbMeas(9)-45; %get a line of brg msmt 80 min
                    %before 9th r/b msmt

hit = [0; rbMeas([1:4]); lob1;rbMeas([5:8]);lob2;rbMeas([9:(nmeas-3)])];
% "hit" is the vector of sample
% intervals
hit = round(hit); %measmt intervals in integer mins

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Track Initialization%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initialize filter 1 with 2 measurements
deltaInit = (90 + 4*randn(1)); %sample interval for the

```

```

                                %two msmts used in init

[xhat(:,1) phat xi] = kal2init(xi, H, Q, sigma1r, sigma1b, deltaInit, s1p);
phatPack(:,1) = reshape(phat,16,1);    %"phatPack" packages the
                                         %covariance matrix
                                         %into a column vector

%initialize filter 2 with 2 measurements
[xhat(:,2) phat2 xi2] = kal2init(xi2, H, Q2, sigma1r, sigma1b, deltaInit, s1p);
phatPack(:,2) = reshape(phat2,16,1);

%Initialization for tracker and Output Matrices for track 1
zout = [];                        %Cartesian msmts
posout = [];                      %true cartesian positions
error = [];                      %cartesian measurement error (from truth)
time = 0;                        %sample times
zest = NaN*ones(4,1);            %position estimate (H*xhat(k|k))
x = xi;                          %state initialization
ind = 1                          %sample index
xb = [];                        %x-coord for LOB plotting
yb = [];                        %y-coord for LOB plotting
xestEll = [];                   %x-coord for corrected estimate
                                %(zest) 95% ellipses
yestEll = [];                   %y-coord for corrected estimate
                                %(zest) 95% ellipses
xPredEll = [];                  %x-coord for predicted estimate
                                %(H*xPred) 95% ellipses
yPredEll = [];                  %y-coord for predicted estimate
                                %(H*xPred) 95% ellipses
estError = [];                  %estimate error (from truth)
y = [];                         %feature measurement ([freq;prf])
clutOut = [];                   %clutter cartesian positions
yClutOut = [];                  %clutter freature measurements

lam = zeros(2);                 %track score and last update time
                                %[score;updTime]
nTracks=2;                      %initial number of tracks
valTrack = [1 2];               %initial vector of valid track numbers
trkCand = [1 2];                %initial vector of track candidates

%filter 2 initialization
zout2 = [];
posout2 = [];
error2 = [];

```

```

time = 0;
zest2 = [];
x2 = xi2;
xe2 = [];
ye2 = [];
xb2 = [];
yb2 = [];
xestEll2 = [];
yestEll2 = [];
xPredEll2 = [];
yPredEll2 = [];
estError2 = [];
y2 = [];
clutOut2 = [];
yClutOut2 = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%tracker simulation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Loop through the simulation, generating target motion between
%sample times and taking measurements at hit times

for ii = 1:length(hit),
    ii
    if (nTracks > 50) break;end;          %stop sim if capacity exceeds 50 tracks

    %*****Target Motion / Filter Prediction*****
    delta = hit(ii);                    %time interval since last measurement:

    % Define the F Matrix (Transition Matrix) for discrete time
    % target motion with constant velocity
    F = [1      delta  0      0;
         0      1      0      0;
         0      0      1      delta;
         0      0      0      1];

    Q = qsqr*[delta^3/3  delta^2/2  0      0;
              delta^2/2  delta      0      0;
              0          0          delta^3/3  delta^2/2;
              0          0          delta^2/2  delta];

    %filter prediction
    for trkNo = 1:nTracks
        phat = reshape(phatPack(:,trkNo),4,4);
        xpred(:,trkNo)=F*xhat(:,trkNo);
    end
end

```



```

    Ppred=F*phat*F' + Q;
    PpredPack(:,trkNo) = reshape(Ppred,16,1);
end

%obtain 3-sigma ellipses around estimate for display purposes
if (ii>5)
    Ppred=reshape(PpredPack(:,1),4,4);
    K = [Ppred(1,1) Ppred(1,3);
         Ppred(3,1) Ppred(3,3)];
    [xell, yell, smaj, smin, theta] = elipa(K, c, xpred(1,1), xpred(3,1));
    xPredEll = [xPredEll xell'];
    yPredEll = [yPredEll yell'];

    Ppred=reshape(PpredPack(:,2),4,4);
    K = [Ppred(1,1) Ppred(1,3);
         Ppred(3,1) Ppred(3,3)];
    [xell, yell, smaj, smin, theta] = elipa(K, c, xpred(1,2), xpred(3,2));
    xPredEll2 = [xPredEll2 xell'];
    yPredEll2 = [yPredEll2 yell'];
end

%target 1 motion step:  $x(k+1) = F*x(k)$ 
%target changes course to 180 at sample time before turnTime
if (time(length(time)) > turnTime)
    x(2) = 0;
    x(4) = -speed;
end
x = F*x;

%target 2 motion step:  $x(k+1) = F*x(k)$ 
%target changes course to 180 at sample time before turnTime
if (time(length(time)) > turnTime2)
    x2(2) = 0;
    x2(4) = -speed2;
end
x2 = F*x2;

%take measmts at hit times:
%take measurement from current target state vector and append
%to the true target position output matrix

```

```

time = [time time(length(time))+hit(ii)];
ztrue = H*x;
posout = [posout      ztrue];

ztrue2 = H*x2;
posout2 = [posout2    ztrue2];
%*****End of Target Motion / Filter Prediction*****

%*****Measurement Generation*****
%obtain noisy msmt from sensor 1 with associated covariance matrix
%in cartesian coords.
%The "if" statement is used to simulate randomly missed measurements
if (rand(1) < 1),
sigR = sigma1r;
sigB = sigma1b;
sigma = [sigR; sigB];                                %packaging

%find difference in positions between sensor and tgt
diff = ztrue - s1p;
diff2 = ztrue2 - s1p;

%target 1 true bearing and range
r = sqrt(diff(1)^2 + diff(2)^2);
brg = atan2(diff(1),diff(2));

%target 2 true bearing and range
r2 = sqrt(diff2(1)^2 + diff2(2)^2);
brg2 = atan2(diff2(1),diff2(2));

%target 1 attribute measurements
fMeas = fTrue + randn(1)*sigmaF;
prfMeas = prfTrue + randn(1)*sigmaPrf;

y = [y [fMeas;prfMeas]];

%target 2 attribute measurements
fMeas2 = fTrue2 + randn(1)*sigmaF;
prfMeas2 = prfTrue2 + randn(1)*sigmaPrf;

y2 = [y2 [fMeas2;prfMeas2]];

```

```

repTyp2d = ~((ii==5)|(ii==10));           %specify msmts to be LOB
                                           %here, 5th and 10th msmt
                                           %will be LOB,
                                           %all others cartesian

if repTyp2d,    %bearing and range msmt
    [z(:,1), R] = get2d(r, brg, sigma, s1p); %tgt 1
                                           %returns noisy msmt and
                                           %msmt covariance

    RPack(:,1) = reshape(R,4,1);
    ztilde = ztrue - z(:,1);
    disterror = sqrt(ztilde'*ztilde); %msmt error

    [z(:,2), R2] = get2d(r2, brg2, sigma, s1p); %tgt 2
                                           %returns noisy msmt and
                                           %msmt covariance

    RPack(:,2) = reshape(R2,4,1);
    ztilde2 = ztrue2 - z(:,2);
    disterror2 = sqrt(ztilde2'*ztilde2); %msmt error

    %add the measurement to the measurement output matrix
    zout = [zout  z(:,1)];
    zout2 = [zout2 z(:,2)];

%This block will produce clutter posit at attribute msmts
if (ii > 5)
    Ppred=reshape(PpredPack(:,1),4,4);
    S1 = H*Ppred*H' + R;
    [clut, nClutter1] = getClut(S1, ztrue, clutDens, gamma);

    %obtains clutter posit msmts
    %uniformly dist around tgt 1
    yClut = getClutFeature(nClutter1, fClut, prfClut, sigmaFClut,
                           sigmaPrfClut); %obtains clutter feature msmts

    clutOut = [clutOut clut];
    yClutOut = [yClutOut yClut];

    Ppred=reshape(PpredPack(:,2),4,4);
    S2 = H*Ppred*H' + R2;
    [clut2, nClutter2] = getClut(S2, ztrue2, clutDens, gamma);

```

```

        yClut2 = getClutFeature(nClutter2, fClut, prfClut, sigmaFClut,
                               sigmaPrfClut);

        clutOut2 = [clutOut2 clut2];
        yClutOut2 = [yClutOut2 yClut2];

    else                                     %ii<5 ==> do not generate clutter msmts
        nClutter1 = 0;
        nClutter2 = 0;
        clut = [];
        clut2 = [];
        yClut = [];
        yClut2 = [];
    end                                     %clutter msmts for 2d case

else                                       %lob msmt

    [z(:,1), xLob, yLob] = getLob(r, brg, sigB, s1p);
                                     %tgt 1 noisy bearing,
                                     %and coordinates for
                                     %plotting LOB

    [z(:,2), xLob2, yLob2] = getLob(r2, brg2, sigB, s1p); %tgt 2

    %add the LOB measurement to the output LOB measurement matrix
    xb = [xb xLob'];
    yb = [yb yLob'];
    xb2 = [xb2 xLob2']; yb2 = [yb2 yLob2'];
    ztilde = nan;                        %msmt distance errors not computed
                                     % for LOB

    disterror = nan;
    ztilde2 = nan;
    disterror2 = nan;
    theta = brg;

end   %measurement generation

%*****End of Measurement Generation*****

    %add the measurement distance error to measurement distance error matrix
    error = [error  disterror];
    error2 = [error2  disterror2];

```

```

%*****Track Processing / Filter Correction*****
%Target Clutter Rejection: Select correct measurement from vector of
%measurements by examining attributes nClutter = nClutter1+nClutter2;
posVec = [z(:,1); z(:,2); clut([1:2*nClutter1])';clut2([1:2*nClutter2])'];
%vector of msmt positions
%note that the correct msmt
%for target 1 is the 1st
%msmt in posVec. Target
%2 is second. All others
%are clutter

attVec = [fMeas; prfMeas; fMeas2; prfMeas2; reshape(yClut,2*nClutter1,1);
          reshape(yClut2,2*nClutter2,1)]; %vector of msmt attributes

for trkNo = valTrack %loops through all tracks
    nVal = 0; %initialize valid msmt counter

    Ppred = reshape(PpredPack(:,trkNo),4,4);
    %extract Ppred for track being
    %processed from packed matrix

    for l = 1:(nClutter+2),
        att = [attVec(2*l-1);attVec(2*l)];
        dT = (att - attTrack(:,trkNo))'*sigT*(att - attTrack(:,trkNo));
        dC = (att - attClut)'*sigC*(att - attClut);
        dTdC(l) = dT/dC;
    end
    %this loop compares the distance of
    %a msmts features to the tgt and
    %clutter means. dTdC is a ratio of
    %these distances.

    valAtt = find(dTdC < gamma); %valAtt holds the index of all msmts
    %that fall below the threshold.
    %These msmts have attributes that
    %are valid for this tgt

    dTdC=[]; %reset for next

%Geometric Track Association and Filter Update: associate observation that
%gives lowest score for this track.
%Track scoring determined by log-likelihood function

%!!!!!!!!!!!!2 cases!!!!!!!!!!!!!!

```

%CASE 1: 2 Dimensional reports using standard filter

%CASE 2: LOB report using EKF

```
xhatSplit(:,1)=[lam(1,trkNo)+6; xpred(:,trkNo); attTrack(2*trkNo-1);  
attTrack(2*trkNo)];
```

%xhatSplit is a holding vector that will contain
%the score, state vector, and attributes that result
%from all measurement associations with the track
%being processed. Here, it is initialized with the
%tracks score, predicted state, and attributes.

```
phatSplitPack(:,1) = [lam(1,trkNo)+6; PpredPack(:,trkNo)];
```

%likewise, phatSplitPack holds the covariances
%resulting from an association

if repTyp2d

%2d (cartesian) report

```
R = reshape(RPack(:,trkNo),2,2); %extract sensor covariance
```

```
S = H*Ppred*H' + R; %S is the innovation covar
```

```
for l = 1:length(valAtt)
```

```
zObs(:,l) = [posVec(2*valAtt(l)-1);posVec(2*valAtt(l))];
```

%obtain a valid (passed the attribute test) msmt

```
ztil = zObs(:,l) - H*xpred(:,trkNo); %the innovation
```

```
lamSplit = ztil'*inv(S)*ztil; %norm of innovation
```

%squared, a distance

%measure

```
if (lamSplit < gamma) %if the dist is inside gate
```

```
lam(2,trkNo) = time(length(time));
```

%indicate time of update in

%score matrix

```
nVal = nVal + 1; %indicate # of valid associat
```

```
G = Ppred*H'*inv(S); %filter gain
```

```
xhatSplit(:,nVal) = [lam(1,trkNo)+lamSplit;
```

```
(xpred(:,trkNo) + G*ztil);
```

```
attVec(2*valAtt(l)-1);
```

```
attVec(2*valAtt(l))];
```

%now, xhatSplit holds new score and

%corrected est

```

        phatSplit = (eye(size(Ppred))-G*H)* Ppred
                *(eye(size(Ppred))-G*H)'+G*R*G';
        phatSplitPack(:,nVal) = [lam(1,trkNo)+lamSplit;
                reshape(phatSplit,16,1)];
        end %valid msmt assoc
end %2d report association

else    %LOB geom association and update

        R = sigB;                                %sensor covar

        x1 = xpred(1,trkNo)-s1p(1);
        x3 = xpred(3,trkNo)-s1p(2);
        den = x1^2 + x3^2;

        h = [-x3/den    0        x1/den        0];
                %linearized msmt matrix

        zhat = atan2(x1,x3);                    %msmt est

        S = h*Ppred*h' + R;                    %innovation covar

        for l = 1:length(valAtt)
                zObs(:,l) = [posVec(2*valAtt(l)-1); posVec(2*valAtt(l))];
                        %obtain a valid observation

                ztil = zObs(1,l) - zhat; %innovation
                lamSplit = ztil'*inv(S)*ztil; %norm of innovation
                        %squared

                if (lamSplit < 4)                %if inside gate...
                        lam(2,trkNo) = time(length(time));
                                %indicate time of update
                                %in score matrix
                        nVal = nVal + 1;%indicate # of valid assoc

                        G = Ppred*h'*inv(S); %gain
                        xhatSplit(:,nVal) = [lam(1,trkNo)+lamSplit;
                                (xpred(:,trkNo) + G*ztil);
                                attVec(2*valAtt(l)-1);
                                attVec(2*valAtt(l))];
                                %now, xhatSplit holds new score
                                %and corrected est

```

```

        phatSplit = (eye(size(Ppred))-G*h) * Ppred *
                    (eye(size(Ppred))-G*h)' + G*R*G';

        phatSplitPack(:,nVal) = [lam(1,trkNo)+
                                lamSplit;
                                reshape(phatSplit,16,1)];

    end
end
%LOB geometric assoc and update

%%%
if trkNo==1
    valAtt1 = length(valAtt);
    nVal1 = nVal;
elseif trkNo==2
    valAtt2 = length(valAtt);
    nVal2 = nVal;
end

%this IF block obtains the no
%of msmts that pass the
%feature clutter rej-
%ection and geographic gates
%for tracks 1 and 2.

%These values will be passed
%to the matrix STAT
%for examination in the
%command window after
%simulation is concluded

%after simulation, this info can be reviewed in the command
>window. For trk 1, the index "1" should be in the valAtt vector. If
%not, attThresh may be too high. Likewise, index "2" should be included
%in valAtt for trkNo 2.

xhatSplit = sortrows(xhatSplit'); %sorts validated estimates into
attSplit = xhatSplit(:,[6:7])'; %ascending order of likelihood
                                %function. attSplit holds the
                                %attribute msmt

phatSplitPack = sortrows(phatSplitPack'); %sorting by likelihood
                                           %here keeps covariances
                                           %indexed with estimates

xhat(:,trkNo) = xhatSplit(1,[2:5])';
phatPack(:,trkNo) = phatSplitPack(1,[2:17])';
lam(1,trkNo) = xhatSplit(1,1);
%lowest score is chosen for tgt track.
%Estimate, Covar, and score are
%updated.

```



```

    for l = 2:nVal                %all remaining associations
        d = size(xhat);           %initialize new tracks.
        xhat(:,d(2)+1)=xhatSplit(1,[2:5])'; %...new trk state est
        phatPack(:,d(2)+1)=phatSplitPack(1,[2:17])'; %...new trk
                                           %est covar
        lam(1,d(2)+1)=xhatSplit(1,1); %...new trk score
        lam(2,d(2)+1)=time(length(time)); %...new trk init time
        attTrack(:,d(2)+1)=attSplit(:,1); %...new trk att
        RPack(:,d(2)+1) = RPack(:,trkNo); %...new trk sensor
                                           %covar
        zest(2*d(2)+1,:) = nan; %...new track in estimate
        zest(2*d(2)+2,:) = nan; %output matrix
        trkCand(d(2)+1) = trkCand(trkNo);
        valTrack = [valTrack d(2)+1];
    end
    xhatSplit = []; %returns this to a column vector
    phatSplitPack = [];
end %trk processing loop

%%%%%%Track Management Algorithm: 2 functions%%%%%
%1) Drop Tracks that exceed thresholds (score and last update).
%2) Select candidate with lowest score to be target estimate
%After all tracks have been processed for a sample time...
%track dropping...
delTrk = find((lam(1,:)>maxScore) | (time(length(time))-lam(2,:))>maxTime);
                                           %form a vector of track numbers for track scores
                                           %that exceed maxScore or update times that exceed
                                           %maxTime
if delTrk
    for l = delTrk
        valTrack(find(valTrack==l)) = [];
        %deletes track from vector of
    end %valid tracks
end

xhat(:,delTrk)=nan;
phatPack(:,delTrk)=nan; %Not-a-number indicates "track deleted"
lam(:,delTrk)=nan;

%best track selection...
%this gets the track number of the candidate w/ lowest score
new1 = floor(find(lam==min(lam(1,find(trkCand==1))))/2)+1;
new2 = floor(find(lam==min(lam(1,find(trkCand==2))))/2)+1;

```

```

%these lines swap the state vector and covariance matrices
%to put the "new" (lower score) vectors into trkNo 1 position.
oldxhat=xhat(:,1);
oldphat=phatPack(:,1);
oldlam=lam(:,1);

xhat(:,1) = xhat(:,new1);
phatPack(:,1) = phatPack(:,new1);
lam(:,1) = lam(:,new1);

xhat(:,new1) = oldxhat;
phatPack(:,new1) = oldphat;
lam(:,new1) = oldlam;

%...likewise for trkNo 2
oldxhat=xhat(:,2);
oldphat=phatPack(:,2);
oldlam=lam(:,2);

xhat(:,2) = xhat(:,new2);
phatPack(:,2) = phatPack(:,new2);
lam(:,2) = lam(:,new2);

xhat(:,new2) = oldxhat;
phatPack(:,new2) = oldphat;
lam(:,new2) = oldlam;

else          %missed msmt
    error = [error nan];
end

%*****form output matrices*****
d = size(xhat);
nTracks = d(2);          %# columns in xhat is # of tracks held

%add estimate to the matrix of estimates
for l=1:nTracks
    ze(:,l) = H*xhat(:,l);
end

zest = [zest reshape(ze,2*1,1)];

```

```

%obtain target 1 error ellipse of estimate
phat = reshape(phatPack(:,1),4,4);
K = [phat(1,1) phat(1,3);
     phat(3,1) phat(3,3)];
[xell, yell, smaj, smin, theta] = elipa(K, c, xhat(1,1), xhat(3,1));
xestEll = [xestEll xell'];
yestEll = [yestEll yell'];
ze=[zest(1,length(zest(1,:))); zest(2,length(zest(2,:)))];
estError = [estError sqrt((ztrue-ze)'*(ztrue-ze))];

```

```

%obtain target 2 error ellipse of estimate
phat2 = reshape(phatPack(:,2),4,4);
K = [phat2(1,1) phat2(1,3);
     phat2(3,1) phat2(3,3)];
[xell, yell, smaj, smin, theta] = elipa(K, c, xhat(1,2), xhat(3,2));
xestEll2 = [xestEll2 xell'];
yestEll2 = [yestEll2 yell'];
ze=[zest(3,length(zest(3,:))); zest(4,length(zest(4,:)))];
estError2 = [estError2 sqrt((ztrue2-ze)'*(ztrue2-ze))];

```

```

%compile statistics for output

```

```

stat(1,ii)=ii;
stat(2,ii)=repTyp2d;
stat(3,ii)=nClutter;
stat(4,ii)=length(valTrack);
stat(5,ii)=nan;
stat(6,ii)=valAtt1;
stat(7,ii)=nVal1;
stat(8,ii)=disterror;
stat(9,ii)=estError(length(estError));
stat(10,ii)=new1;
stat(11,ii)=length(find(trkCand(valTrack)==1));
stat(12,ii)=nan;
stat(13,ii)=valAtt2;
stat(14,ii)=nVal2;
stat(15,ii)=disterror2;
stat(16,ii)=estError2(length(estError2));
stat(17,ii)=new2;
stat(18,ii)=length(find(trkCand(valTrack)==2));

```

```

%the matrix STAT will be displayed in the
%command window for post-simulation
%performance analysis

```

```

end
%time step
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Output Results%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time(1) = [];

```

```

%Plot the results
nEst = nmeas;
nmeas = max(size(zout));

%Target motion and true position at measurement times
figure(1)
axis equal
plot(posout(1,:),posout(2,:),'-',posout2(1,:),posout2(2,:),'-',s1p(1),s1p(2),'X');grid;
title('True Target Motion');
xlabel('Deg Lon');ylabel('Deg Lat')

%Target motion with noisy measurements
figure(2)
nbrg = min(size(xb));
nPred = min(size(xPredEll));
plot(posout(1,:),posout(2,:), 'o',zout(1,:),zout(2,:), '*',xb(:,[1:nbrg]),yb(:,[1:nbrg]),
    xb2(:,[1:nbrg]),yb2(:,[1:nbrg]),clutOut(1,:),clutOut(2,:), '*',s1p(1),s1p(2), 'X');
hold on
plot(posout2(1,:),posout2(2,:), 'o',zout2(1,:),zout2(2,:), '*',clutOut2(1,:),clutOut2(2,:), '*',
    s1p(1),s1p(2), 'X',xPredEll(:,[1:nPred]),yPredEll(:,[1:nPred]),
    xPredEll2(:,[1:nPred]),yPredEll2(:,[1:nPred]),s1p(1),s1p(2), 'X');grid;
axis equal
title('Target Motion and Measurements at Sample Times');
xlabel('Deg Lon');ylabel('Deg Lat')
hold off

%Target motion with Filter Estimates
figure(3)
plot(posout(1,:),posout(2,:), 'o',zout(1,:),zout(2,:), '*',xb(:,[1:nbrg]),yb(:,[1:nbrg]),
    xb2(:,[1:nbrg]),yb2(:,[1:nbrg]),zest(2*[1:nTracks]-1,:),zest(2*[1:nTracks],:),'x',
    xestEll(:,[1:nEst]),yestEll(:,[1:nEst]),s1p(1),s1p(2), 'X');grid;
hold on
plot(posout2(1,:),posout2(2,:), 'o',zout2(1,:),zout2(2,:), '*',
    xestEll2(:,[1:nEst]),yestEll2(:,[1:nEst]),s1p(1),s1p(2), 'X');grid;
axis equal
title('Target Motion, Measurements, and Estimates at Sample Times');
xlabel('Deg Lon');ylabel('Deg Lat')
hold off

%Target 1 motion with Filter Estimates
figure(4)
plot(posout(1,:),posout(2,:), 'o',zout(1,:),zout(2,:), '*',xb(:,[1:nbrg]),yb(:,[1:nbrg]),
    zest(1,:),zest(2,:), 'x',xestEll(:,[1:nEst]),yestEll(:,[1:nEst]),s1p(1),s1p(2), 'X');grid;

```

```

axis equal
title('Target 1 Motion, Measurements, and Estimates at Sample Times');
xlabel('Deg Lon');ylabel('Deg Lat')

%Target 2 motion with Filter Estimates
figure(5)
plot(posout2(1,:),posout2(2,:), 'o', zout2(1,:),zout2(2,:), '*',xb2(:,[1:nbrg]),yb2(:,[1:nbrg]),ze
    st(3,:),zest(4,:), 'x',xestEll2(:,[1:nEst]),yestEll2(:,[1:nEst]),s1p(1),s1p(2), 'X'); grid;
axis equal
title('Target 2 Motion, Measurements, and Estimates at Sample Times');
xlabel('Deg Lon');ylabel('Deg Lat')

stat

```

kal2init.m

```

function [xhat, phat, xiNew] = kal2init(xi, H, Q, sigR, sigB, delta, sensPos)
%initializes the kalman filter based on 2 measurements
%
%The 2 "true" positions are xi and F*xi.
%These are converted to noisy measurements using pol2cart. The associated
%cartesian covariance matrices are also obtained.

%From these noisy measurements and covariances, the initial state and covariance
%for the filter is estimated.

%INPUT parameters:
%   xi:          true target state at time zero
%   H:           Measurement matrix
%   Q:           Plant Noise
%   sigR:        Sensor Range Inaccuracy (std dev)
%   sigB:        Sensor Bearing Inaccuracy (std dev)
%   delta:       Sample Interval
%   sensPos:     Sensor Position

%OUTPUT parameters
%   xhat:        Initial State Estimate
%   phat:        Initial State Estimate Covariance
%   xiNew:       new true target state (xi at time delta)

%Supporting m-files:
%   pole2cart.m

```

```

% Define the F Matrix (Transition Matrix) for discrete time
% target motion with constant velocity
F = [1 delta 0 0;
     0 1 0 0;
     0 0 1 delta;
     0 0 0 1];

[z1 R1] = pole2cart(H*xi, sensPos, sigR, sigB);
xiNew = F*xi;
[z2 R2] = pole2cart(H*xiNew, sensPos, sigR, sigB);

z = [z2; z1];

A = [1 0 0 0;
     1/delta 0 -1/delta 0;
     0 1 0 0;
     0 1/delta 0 -1/delta];
invf = inv(F);

xhat = A*z;
phat = A*[R2 zeros(2,2);zeros(2,2) R1+H*invf*Q*invf'*H]*A';

```

pole2cart.m

```

function [z, R, disterror] = pole2cart(ztrue, sp, sigmar, sigmab);

```

```

%obtains a noisy polar measurement to target, adds measurement noise,
%and converts measurement and covariance matrix to cartesian
%inputs:

```

```

%   ztrue = [xt; yt]      true posit of target in cartesian
%   sp     = [xs; ys]     posit of sensor in cartesian
%   sigmar = scalar        range msmt standard deviation
%   sigmab = scalar        bearing msmt standard deviation

```

```

%outputs:

```

```

%   z = [xm; ym]          noisy measurement in cartesian
%   R = 2 X 2             covariance matrix in cartesianj
%   disterror             = sqrt((ztrue - z)*(ztrue - z)')

```

```

sigma = [sigmar; sigmab];

%find difference in positions between sensor and tgt
diff = ztrue - sp;

%true bearing and range
r = sqrt(diff(1)^2 + diff(2)^2);
brg = atan2(diff(1),diff(2));
zplr = [r; brg];

%add noise to the measurement
zmeas = zplr + randn(size(sigma)).*sigma;

%convert the noisy measurement to cartesian
z = [zmeas(1)*sin(zmeas(2));
     zmeas(1)*cos(zmeas(2))];
z = sp + z;

%convert covariance matrix to cartesian
sv = diag(sigma.^2);
M = [sin(zmeas(2)) -zmeas(1)*cos(zmeas(2));
     cos(zmeas(2))  zmeas(1)*sin(zmeas(2))];

R = M*sv*M';

%calculate distance error
ztilde = ztrue - z;
disterror = sqrt(ztilde*ztilde);

```

elipa.m

```
function [xout, yout, smaj, smin, theta] = elipa(PK, c, xt, yt)
```

```

%calculates error ellipsoids given error covariance
%and estimate position

```

```
%INPUTS:
```

```

%   PK is covariance matrix
%   c is confidence region (c=sqrt(6) for 95%)
%   xt is ellipse center x coord
%   yt is ellipse center y coord

```

```

%OUTPUTS:
%   xout is a vector containing x coord points of the ellipse
%   yout is a vector containing y coord points of the ellipse
%   smaj is the length of the semi-major axis
%   smin is the length of the semi-minor axis
%   theta is the orientation of the semi-major axis

% to plot the ellipse, use: plot(xout,yout)

%adapted from Stephen L. Spehn's Errellip.m (15 Nov 89) by Mark Olson

%get eigenvalues (lam) and eigenvectors(V)
[V,lam] = eig(PK);
sigx = sqrt(lam(1,1));
sigy = sqrt(lam(2,2));

%parameterized ellipse
t = 0:2*pi/100:2*pi;
x = sigx*c*cos(t);
y = sigy*c*sin(t);

%translate to eigenvectors space and center at tgt posit
xout = x*V(1,1) + y*V(1,2) + xt;
yout = x*V(2,1) + y*V(2,2) + yt;

%report semimajor and semiminor axes lengths and orientation
smaj = 2*c*max([sigx sigy]);
smin = 2*c*min([sigx sigy]);
theta = atan2(V(2,1),V(1,1)) + pi/2*(sigy > sigx);    %sigy > sigx ==> y is smaj

```

get2d.m

```

function [z, R] = get2d(r, brg, sigma, sensPos)

%this function generates a noisy cartesian measurement from
%true bearing and range. Polar measurement noise is added to
%true polar position, then rotated into a cartesian coordinate
%reference. The measurement covariance is also rotated into a
%cartesian coordinate reference.

```

```

%INPUTS:
%   r:           true range to target

```



```
%    brg:          true bearing to target
%    sigma:        column vector of sensor std dev [sigB;sigR]
%    sensPos:       sensor position in cartesian coordinates
```

```
%OUTPUTS:
```

```
%    z:            cartesian measurement
%    R:            cartesian measurement covariance
```

```
%obtains cartesian measurement true range and brg msmt
```

```
    zplr = [r; brg];
```

```
%add noise to the measurement
```

```
    zmeas = zplr + randn(size(sigma)).*sigma;
```

```
%convert the noisy measurement to cartesian for plotting
```

```
    z = [zmeas(1)*sin(zmeas(2));
          zmeas(1)*cos(zmeas(2))];
    z = sensPos + z;
```

```
%obtain target 1 covariance of error in cartesian
```

```
    M = [sin(zmeas(2))    -zmeas(1)*cos(zmeas(2));
          cos(zmeas(2))    zmeas(1)*sin(zmeas(2))];
```

```
    R = M*diag(sigma.^2)*M';
```

getLob.m

```
function [zbrg ,xLob, yLob] = getLob(r, brg, sigB, slp);
```

```
%this function generates a noisy measurement bearing and output
%vectors for plotting from true target position.
```

```
%INPUTS
```

```
%    r:            true target range
%    brg:          true target bearing
%    sigB:         sensor bearing measurement std dev
%    slp:          sensor position in cartesian coord
```

```
%OUTPUTS
```

```
%    zbrg:         measured bearing
```

```
%    xLob:      Line of Bearing x-coords for plotting
%    yLob:      Line of Bearing y-coords for plotting
```

```
%to plot the Line of Bearing, use
%    plot(xLob,yLob)
```

```
%find Line of Bearing Msmt to target
```

```
%add noise to the measurement
%    zbrg = brg + randn*sigB;
```

```
%convert the noisy measurement to cartesian for plotting
%    xLob = linspace(3,(r+1)*sin(brg));
%    yLob = xLob/tan(zbrg);
%    xLob = slp(1)*ones(size(xLob)) + xLob;
%    yLob = slp(2)*ones(size(xLob)) + yLob;
```

getClut.m

```
function [clut, nClutter] = getClut(S, ztrue, clutDens, gamma);
```

```
%this function generates uniformly distributed clutter position measurement
%in a region around the true target position. (Bar Shalom and Li, 1995)
```

```
%INPUTS:
```

```
%    S:          Innovation Covariance  $S = H*P*H' + R$ 
%    ztrue:       true cartesian position of target, [x-coord;y-coord]
%    clutDens:     Clutter Density, tgts/area (here, tgts/3600 sq nm)
%    gamma:       chi-squared dist threshold corresponding to validation region
%                 size. (here, gamma=sqrt(6) for 95% probability ellipse
```

```
%OUTPUTS:
```

```
%    clut:        matrix of clutter positions clut = [x-coord of all clutter msmts;
%                                                    y-coord of all clutter msmts]
%    nClutter:     number of clutter msmts generated
```

```
%obtain clutter posit measmts
```

```
%generate target 1 clutter measurements: uniformly distributed in a
```

%window around the true target position:

%determine the area of the validation region (ref: YBS MMT sec 3.4.6)

Vk = pi*gamma*sqrt(det(S));

nClutter = round(10*Vk*clutDens+1);%Number of false msmsts in a square
%around truth

xClutter = sqrt(10*Vk)*(rand(1,nClutter) - .5*ones(1,nClutter)) + ztrue(1);

yClutter = sqrt(10*Vk)*(rand(1,nClutter) - .5*ones(1,nClutter)) + ztrue(2);

clut = [xClutter; yClutter];

getClutFeature.m

function yClut = getClutFeature(nClutter, fClut, prfClut, sigmaFClut, sigmaPrfClut);

%this file obtains the feature (emitter freq and prf) measurements for
%the false targets. Normal dist is assumed.

%INPUTS:

%	nClutter:	the number of clutter targets
%	fClut:	mean emitter frequency for clutter targets
%	prfClut:	mean emitter prf for clutter targets
%	sigmaFClut:	std dev of emitter freq for clutter targets
%	sigmaPrfClut:	std dev of emitter prf for clutter targets

%OUTPUT:

%	yClut:	matrix of feature measurement column vectors
%		[emitter freq for all targets;
%		emitter prf for all targets]

fMeasClut = fClut*ones(1,nClutter) + randn(1,nClutter)*sigmaFClut;

prfMeasClut = prfClut*ones(1,nClutter) + randn(1,nClutter)*sigmaPrfClut;

yClut = [fMeasClut;prfMeasClut];

LIST OF REFERENCES

Bar-shalom, Y. and Li, X., *Multitarget-Multisensor Tracking: Principles and Techniques*, Yaakov Bar-Shalom, Storrs, CT, 1995.

Bar-shalom, Y. and Li, X., *Estimation and Tracking: Principles, Techniques, and Software*, Yaakov Bar-Shalom, Storrs, CT, 1998.

Kaplan, W., *Advanced Mathematics for Engineers*, Addison-Wesley Publishing Co, Reading, MA., 1981.

Kirk, D. E., *Optimal Estimation: An Introduction to the Theory and Applications*, Donald E. Kirk, Monterey, CA, 1975.

Spehn, S. L., *Noise Adaptation and Correlated Maneuver Gating of an Extended Kalman Filter*, Master's Thesis, Naval Postgraduate School, California, 1990.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center.....2 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	
2. Dudley Knox Library2 Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	
3. Chairman1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	
4. Engineering and Technology Curriculum Office, Code 34.....1 Department of Engineering and Technology Naval Postgraduate School Monterey, CA 93943-5109	
5. Professor Herschel Loomis, Code EC/Lm.....1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	
6. Professor Harold Titus, Code EC/Ti.....1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	
7. Dr. Alan Ross, Code SP1 Space Systems Academic Group Naval Postgraduate School Monterey, CA 93943-5110	
8. Professor Gary Hutchins, Code EC/Hu1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	

9.	LT Mark Olson.....	1
	1620 Friday Ln	
	Mansfield, OH 44906	